

MacTech Magazine
Vol. 18, No. 11 • November 2002

MacTech®

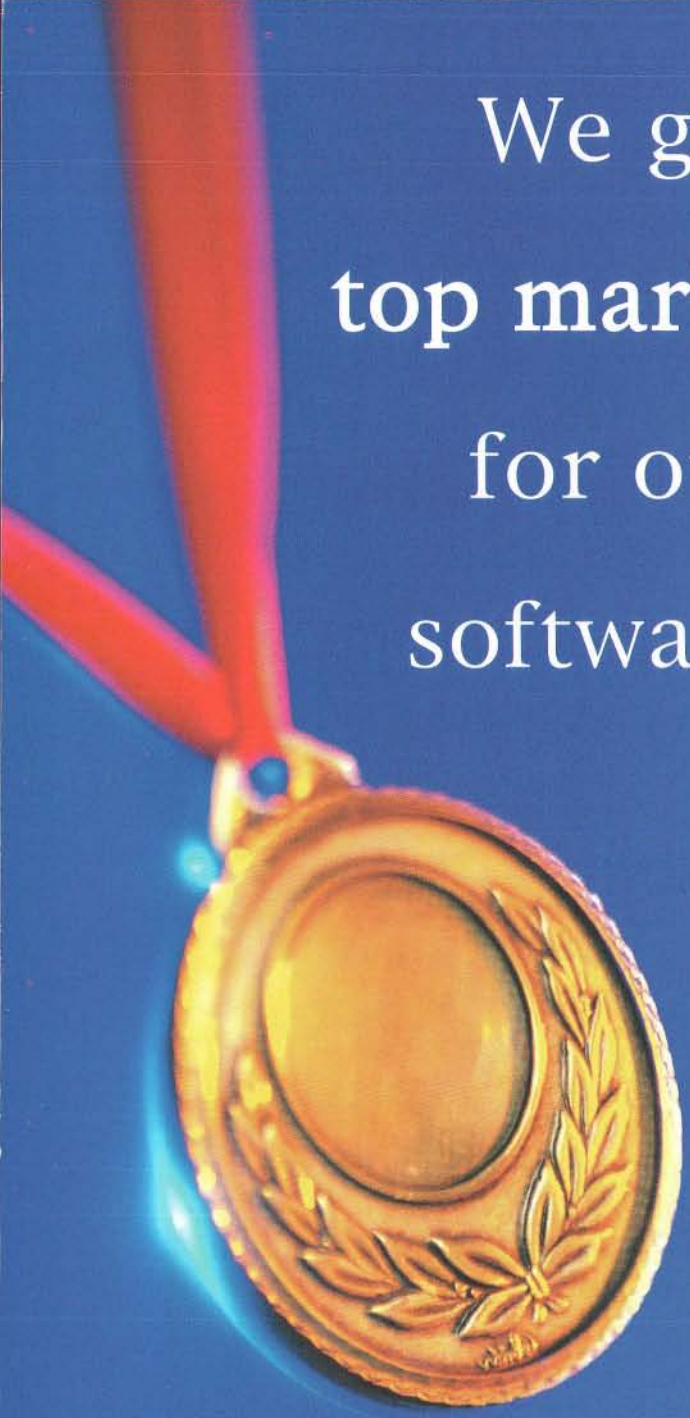
The Journal of Macintosh Technology and Development



GETTING STARTED

\$3.95 US
\$2.95 Canada
SN 1067-8360
Printed in U.S.A.





We got
top marks
for our
software.

You can too.

Revolution™
The Solution for Software Development
In enterprise, business and education

REVOLUTION™

Limitless possibilities

With Revolution, the solutions you can create are endless.

True cross-platform development

With the click of a button, you can build applications for Macintosh (Classic and OS X), Windows, Linux, and Unix.

Increased productivity

The powerful interface builder and easy-to-use programming language speed up all the essentials of software development, leaving you with more time to focus on your project.

Databases, multimedia, Internet applications, external libraries and more

Revolution supports everything you need: SQL databases, streaming media, control of QuickTime, QTVR and graphics, CGI processing, Internet protocols, and more.

FREE Trial Version

www.runrev.com



MacUser UK - 5 mice

©2002 Runtime Revolution Limited. All rights reserved. Runtime Revolution, the Runtime Revolution logo and Revolution are trademarks of Runtime Revolution Limited, registered in the United Kingdom. All other trademarks are the property of their respective owners.

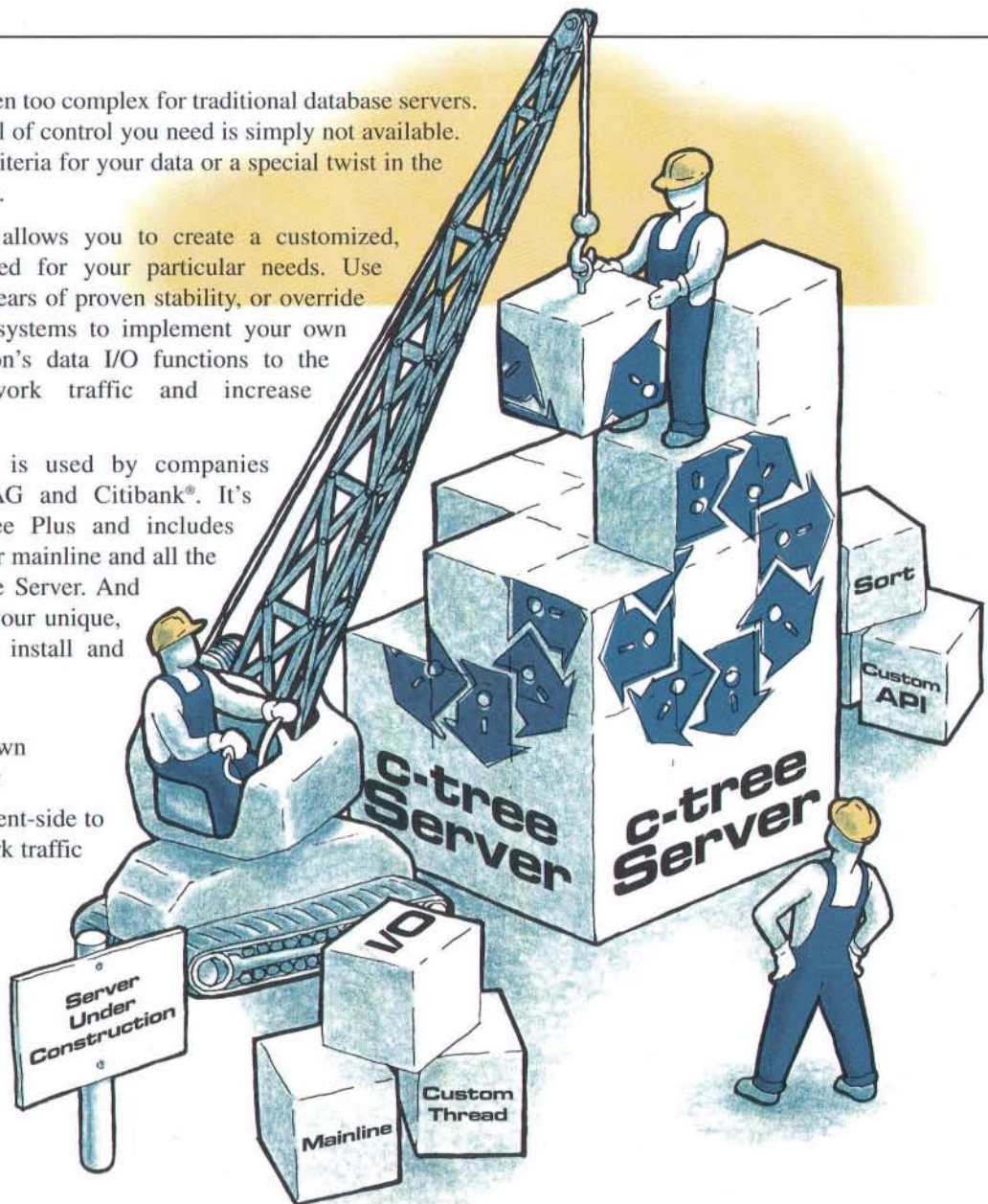
CUSTOMIZE YOUR DATABASE SERVER WITH THE C-TREE® SERVER SDK

Today's database demands are often too complex for traditional database servers. The functionality and precise level of control you need is simply not available. Perhaps you need alternate sort criteria for your data or a special twist in the threading or communication logic.

FairCom's c-tree® Server SDK allows you to create a customized, industrial-strength server designed for your particular needs. Use FairCom's kernel, with over 20 years of proven stability, or override functionality within specific subsystems to implement your own subtleties. Move your application's data I/O functions to the server-side to decrease network traffic and increase performance!

FairCom's c-tree Server SDK is used by companies worldwide such as Software AG and Citibank®. It's integrated seamlessly into c-tree Plus and includes complete source code to the server mainline and all the interface subsystems to the c-tree Server. And best of all, once you've created your unique, customized server, it is easy to install and administer: no DBA required!

- Enhance our server with your own custom server-side functionality
- Move functionality from the client-side to the server-side to reduce network traffic and increase performance
- Modify or replace entire server subsystems
- Complete source for the server mainline, key server subsystems, and client-side
- Flexible OEM licensing



Visit www.faircom.com/ep/mt/sdk today to take control of your server!



FairCom®
www.faircom.com

| | |
|--------|------------------|
| USA | 573.445.6833 |
| EUROPE | +39.035.773.464 |
| JAPAN | +81.59.229.7504 |
| BRAZIL | +55.11.3872.9802 |

DBMS Since 1979 • 800.234.8180 • info@faircom.com

Other company and product names are registered trademarks or trademarks of their respective owners.

© 2002 FairCom Corporation

XSERVE SCREAMS.

(FOR AN ENTERPRISE CLASS DATABASE.)

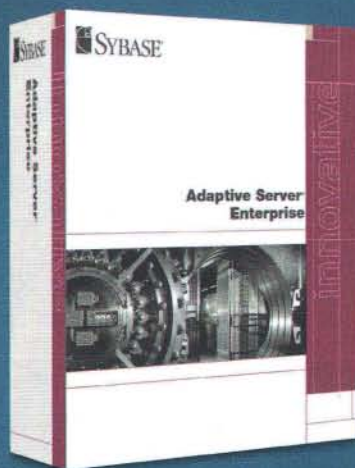


An insanely great enterprise server deserves an insanely great enterprise database. Sybase Adaptive Server™ Enterprise 12.5. Wall Street's preferred platform for mission-critical, transaction-intensive

SYBASE e-BUSINESS SOFTWARE. EVERYTHIN

THE STRAIGHT GOODS ON DATABASES.

WE ANSWER THE CALL.



enterprise applications. Now available to you on Apple Xserve™ with Mac OS X Server software. Ready to scream for joy? Check it out at www.sybase.com/mac



WORKS BETTER WHEN EVERYTHING WORKS TOGETHER.™

©2002 Sybase, Inc. All rights reserved. All trademarks are the property of their respective owners.

How To Communicate With Us

In this electronic age, the art of communication has become both easier and more complicated. Is it any surprise that we prefer **e-mail**?

If you have any questions, feel free to call us at 805/494-9797 or fax us at 805/494-9798.

If you would like a subscription or need customer service, feel free to contact Developer Depot Customer Service at 877-MACTECH

DEPARTMENTS

Orders, Circulation, & Customer Service

Press Releases

Ad Sales

Editorial

Programmer's Challenge

Online Support

Accounting

Marketing

General

Web Site (articles, info, URLs and more...)

E-Mail/URL

cust_service@mactech.com

press_releases@mactech.com

ad_sales@mactech.com

editorial@mactech.com

prog_challenge@mactech.com

online@mactech.com

accounting@mactech.com

marketing@mactech.com

info@mactech.com

<http://www.mactech.com>

The MacTech Editorial Staff

Publisher • Neil Ticktin

Managing Editor • Jessica Stubblefield

Online Editor • Jeff Clites

Regular Columnists

Getting Started

by Dave Mark

Programmer's Challenge

by Bob Boonstra

QuickTime ToolKit

by Tim Monroe

Reviews/KoolTools

by Michael R. Harvey

Regular Contributors

Vicki Brown, Andrew Stone,
Erick Tejkowski, Paul E. Seving

MacTech's Board of Advisors

Jordan Dea-Mattson, Jim Straus
and Jon Wiederspan

MacTech's Contributing Editors

- Michael Brian Bentley
- Marshall Clow
- John. C. Daub
- Tom Djajadiningrat
- Bill Doerrfeld, Blueworld
- Andrew S. Downs
- Richard V. Ford, Packeteer
- Gordon Garb, Cobalt Networks
- Ilene Hoffman
- Chris Kilbourn, Digital Forest
- Rich Morin
- John O'Fallon, Maxum Development
- Will Porter
- Avi Rappoport, Search Tools Consulting
- Ty Shipman, Kagi
- Chuck Shotton, BIAP Systems
- Cal Simone, Main Event Software
- Steve Sisak, Codewell Corporation
- Andrew C. Stone, www.stone.com
- Chuck Von Rospach, Plaidworks
- John C. Welch

Xplain Corporation Staff

Chief Executive Officer • Neil Ticktin

President • Andrea J. Sniderman

Controller • Michael Friedman

Production Manager • Jessica Stubblefield

Production/Layout • Darryl Smart

Marketing Manager • Nick DeMello

Events Manager • Susan M. Worley

Network Administrator • David Breffitt

Accounting • Jan Webber, Marcie Moriarty

Customer Relations • Laura Lane
Susan Pomrantz

Shipping/Receiving • Joel Licardie

Board of Advisors • Steven Geller, Blake Park,
and Alan Carsrud

All contents are Copyright 1984-2002 by Xplain Corporation. All rights reserved. MacTech and Developer Depot are registered trademarks of Xplain Corporation. RadGad, Useful Gifts and Gadgets, Xplain, DevDepot, Depot, The Depot, Depot Store, Video Depot, Movie Depot, Palm Depot, Game Depot, Flashlight Depot, Explain It, MacDev-1, THINK Reference, NetProfessional, NetProLive, JavaTech, WebTech, BeTech, LinuxTech, MacTech Central and the MacTutorMan are trademarks or service marks of Xplain Corporation. Sprocket is a registered trademark of eSprocket Corporation. Other trademarks and copyrights appearing in this printing or software remain the property of their respective holders.

MacTech Magazine (ISSN: 1067-8360 / USPS: 010-227) is published monthly by Xplain Corporation, 850-P Hampshire Road, Westlake Village, CA 91361-2800. Voice: 805/494-9797, FAX: 805/494-9798. Domestic subscription rates are \$47.00 per year. Canadian subscriptions are \$59.00 per year. All other international subscriptions are \$97.00 per year. Domestic source code disk subscriptions are \$77 per year. All international disk subscriptions are \$97.00 a year. Please remit in U.S. funds only. Periodical postage is paid at Thousand Oaks, CA and at additional mailing office.

POSTMASTER: Send address changes to **MacTech Magazine**, P.O. Box 5200, Westlake Village, CA 91359-5200.

C o n t e n t s

November 2002 • Volume 18, Issue 11

MAC OS X

- 8 **Report on the First O'Reilly
Mac OS X Conference**
Or How I spent One Fun-filled Week in October
by Vicki Brown

COVER STORY

- 46 **GETTING STARTED**
Getting Started: Circa 2002
by Dave Mark

MAC OS X

- 54 **The Ins and Outs of Drag and Drop**
by Andrew C. Stone

COCOA DEVELOPEMENT

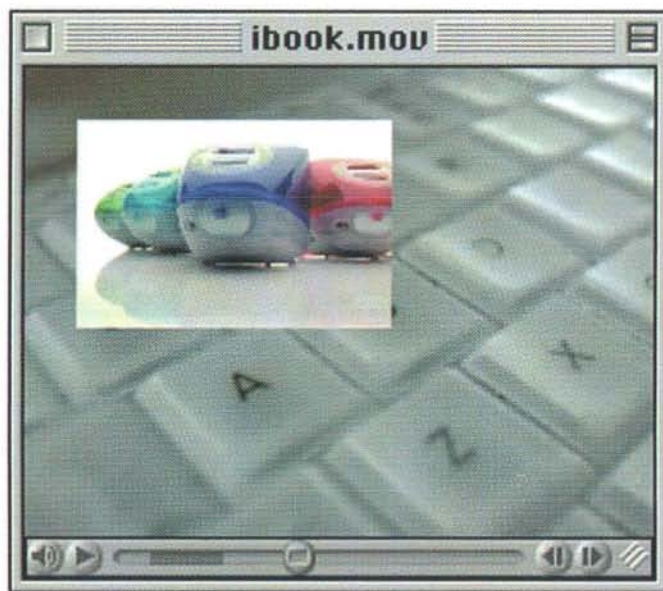
- 64 **Table Techniques Taught
Tastefully (part 3)**
Using NSTableView for Real-World Applications
By Dan Wood, Alameda CA



Drag files onto Dock or Window. page 54

QUICKTIME TOOLKIT

- 24 **She's Gotta Have It**
Using Media Sample References and Data References
by Tim Monroe



A child movie inside of a parent movie. page 32

EDITOR'S DESK

- 6 by Dave Mark, Editor In Chief

PROGRAMMER'S CHALLENGE

- 12 **PENULTIMATE**
by Bob Boonstra

By Dave Mark, Editor-in-Chief

FIRST, A NOTE FROM THE PUBLISHER'S DESK

It's not often that I get to write, but I did want to touch base with you, our readers, this month on a couple of notable events here at *MacTech Magazine*.

First, as you will see later in this issue, we're winding down the Programmer's Challenge. As the Challenge is now 10 years old, Bob wants to spend more time with his family. We wish Bob all the best, and give him a hearty "thank you" for all the efforts he's made in helping us learn to program more efficiently.

We are considering new kinds of puzzles and contents, what's doable, and what's not. If you have ideas or feedback on what you'd like to see, drop us a line.

HELLO (AGAIN) DAVE!

As you probably already heard, and see below, we welcome back Dave Mark with open arms. Dave is coming back not only with the "Getting Started" column, but as our new Editor-in-Chief.

We're thrilled to have Dave on board again. We'll continue to cover the kinds of things you want, from REALbasic to CodeWarrior, from Scripting to Network Administration topics. And, we'll look to have even more fun along the way. We're looking for feedback from you all!

Neil Ticktin, Publisher

GOOD TO BE BACK

Ten years ago, before the net was big, before Rhapsody, before Copland, even before Greg Galanos and company begat CodeWarrior, I had a chance encounter with my good friend (and your faithful publisher) Neil Ticktin. If memory serves, Neil was at MacWorld having just put the finishing touches on his deal to purchase a then struggling magazine, known as MacTutor, soon to become MacTech. Neil was a blizzard of activity, signing this and doing that, and I got sucked into the maelstrom.

For those young pups in the crowd, I used to be a regular in MacTech. Wrote a column called Getting Started, and another called "From the Factory Floor".

Wrote a bunch of books as well. If you don't know me, maybe you've stumbled across a well-worn copy of "Learn C on the Macintosh" or one of the "The Macintosh Programming Primer" series.

Since my days with MacTech, I've done a lot of adventuring. I hooked up with Metrowerks, had a lot of fun there, met some awesome people. Metrowerks up and sold themselves to Motorola, time to go, started up another startup with some friends (buy me a drink at the next MacWorld or WWDC and I'll tell you all about that roller-coaster ride!)

Somewhere about a year ago, the events started unfolding that brought me back here.

Way back when, I'd played with all the Rhapsody betas, run the early Mac ports of Project Builder and Interface Builder, messed around with Objective-C. But, I don't know, none of it really rang true for me.

Then, about a year ago, a friend of mine gave me his cube, just so I could play with the Mac OS X beta. I was skeptical. And then Apple shipped the Titanium PowerBooks. And I fell in love all over again.

So I'm out in California visiting my buddy Neil, and he plants the seed. "Dave". "Dave". "Come back, Dave". "Write more books". "Start writing your Getting Started column again". "I'll even make you Editor-in-Chief, Dave". LOL. Well, maybe it wasn't quite like that. But close. :)

So here I am. Back in the fold. Enjoying the heck out of my Mac. Doing a tremendous amount of learning. There is SO much cool stuff to play with now. Cool NEW stuff. Like Cocoa, Objective-C, and Objective-C++. Amazing Java and REALbasic apps that feel totally natural running under X. My old friends AppleScript and QuickTime are still here and incredibly well integrated into this new environment.

On the hardware side, Apple has really hit their stride. The digital hub strategy makes sense to me. There's a gorgeous, focused lineup of machines, each outdoing its predecessor. USB and FireWire are also well integrated and ubiquitous enough in the consumer electronics world to make the digital hub a realistic strategy. And the iPod? The iPod is like the icing on the cake.

Bottom line, there's a lot of cool stuff happening and I am really having a blast playing with all these cool toys and learning to program my Mac all over again.

As you can see if you turn the page a bit, I've started my monthly Getting Started column once again. But this time around, my involvement with MacTech is going to be quite a bit deeper. Part of my role as Editor-in-Chief is to reshape the magazine, to find ways to make MacTech Magazine more useful to you.

To that end, we've created an email address that goes right to my doorstep: feedback@mactech.com

Please take a minute or two to drop me a line. Let me know what you like and dislike about the magazine. Want more OS X-centric content? More on Carbon? How about topics like PHP and MySQL? Though I can't promise to reply to every email, I will promise to read every one of them.

All that said, it is great to be back in the fold. I am really looking forward to working with you all again.

*From the Editor's chair...
Dave Mark*

Borland®

#1 in Java™
Development
Solutions

Borland delivers leading Java application development technologies for building, optimizing, and deploying business-critical J2EE™ platform applications and Web Services.

Maximize the benefits of your enterprise Java solutions. From development to deployment, the award-winning Borland® software platform for Java enables you to increase productivity and performance while lowering the total

cost of ownership and reducing time-to-market. The result is strong, reliable applications that take full advantage of the power of Java.

Discover the key to success with Java technologies from the industry leader: Borland. Get your FREE *Enterprise Guide to Java Development* at info.borland.com/new/javasolutions/92125.html

Borland®

By Vicki Brown

Report on the 1st O'Reilly Mac OS X Conference

IMAGINE...

Imagine the introductory session of a technical conference. Approximately seven hundred developers and users watch and listen as the speaker welcomes them to four days of interesting talks given by experts in the field. The room is softly lit by the screens of hundreds of laptop computers, about one for every two attendees. The majority of the laptops are white iBooks and Titanium Powerbooks, with a sprinkling of older Powerbooks and iBooks here and there. Every screen you see is running Mac OS X.

The session rooms and the break area are provided with open wireless connectivity, both between systems at the conference and to the Internet. Every session room is outfitted with Macintosh hardware and a Cinema display, but most of the speakers bring their own Powerbooks. Connecting the Powerbooks to the projection facilities is easily done; the tech crew all understand Macintosh.

Attendees scan the program. Sessions this week include keynote sessions by David Pogue (NY Times Technology Columnist and Mac Author/Publisher), James Gosling (V.P. and Fellow at Sun, co-inventor of Java, etc.), Dan Gillmor (San Jose Mercury News Technology Columnist, Jordan Hubbard (Manager of BSD Technologies, Apple Computer), Wilfredo Sanchez Vega (Darwin Developer) and Mark Fruenfelder (writer and illustrator).

Technical sessions cover Cocoa programming, Aqua, Quartz, QuickTime, Open Source and Darwin, Java, WebObjects, Mac OS X Server, iPhoto, Rendezvous, Apple Help, and AppleScript. Less technical sessions discuss end-user troubleshooting, what's new in Jaguar, an overview of Mac OS X for Mac OS 9 users, and a Mac OS X "report card" (presented by Adam Engst, editor of TidBITS).

This isn't Apple's World Wide Developer Conference, but it is definitely a developer-oriented conference. Nor is it MacHack; there is hackery, to be sure, but it mostly happens during the daylight hours. And, of course, this isn't MacWorld; the tiny exhibit space and strong technical focus both emphasize that fact.

In short, this is the first annual Mac OS X Conference, held by O'Reilly & Associates from Sept. 30 - Oct. 3, 2002 in Santa Clara, CA. O'Reilly & Associates, a well-known publisher of books on Unix and open source topics such as Perl, Python, Linux, Apache, and many others, has also become known as a presenter of excellent technical conferences. Notable O'Reilly offerings have included several Open Source conventions, the Emerging Technologies conference, as well as conferences on Bioinformatics, Java, and Peer-to-Peer services.

Although Tim O'Reilly (founder and president of O'Reilly & Associates) is not a developer himself, he is an avid follower of technology and technologists. In fact, he claims that much of his business model is based on following the activities of "Alpha Geeks", the techies who always seem to "get there first" on any new and interesting technology.

Originally focusing on Unix-specific topics, O'Reilly & Associates has published a number of books on cross-platform technologies (such as Bioinformatics, Java and XML) as well as publications on the Windows and Macintosh platforms. Mac OS X brings many of O'Reilly's areas of interest together; the first Mac OS X conference was an opportunity for O'Reilly to bring practitioners of those interest areas together to share knowledge and expertise.

INTRODUCTION

The conference program provided this introduction:

Welcome to our first conference focusing on Mac OS X, one of the most visionary yet practical things happening in the industry today. ... Just as programming tools and applications now share common ground in Mac OS X, this conference brings Mac, Unix/open source, Java, and other practitioners into the same space... It's an event designed to speed your transition or introduction to the 21st century operating system.

Tim O'Reilly is also quoted (on the program cover) as saying:

Mac OS X is the first true 21st century platform. The BSD Unix underpinnings bring stability and a rich open source heritage; the Aqua interface brings Apple's longstanding expertise in user experience. The iApps, 802.11 wireless support, and peer-to-peer features like Rendezvous show that Apple hasn't lost its touch when it comes to putting the future of computing into a 'sleek, insanely great' package.

Any developer who isn't tracking Mac OS X ought to have his head examined.

THE SESSIONS

O'Reilly called this a conference for developers, power users, hackers and network administrators, and it definitely was. There was something for each of these groups, and several other categories besides.

Vicki Brown has been using Unix systems since 1983 and Mac OS since 1986. She is delighted that Mac OS X gives her the opportunity to use both at the same time.

MAC OS X IN THE LARGE

For users new to Mac OS X, there were several talks to choose from, including David Pogue's "Welcome to Mac OS X", based on his book "Mac OS X: The Missing Manual" (published by O'Reilly). This session was part of a track entitled "Mac OS X in the Large".

Other sessions in this track included Adam Engst's "Mac OS X Report Card", a session entitled "End-user Troubleshooting for Fun and Profit" (based on the book, "Mac OS X Disaster Relief" by Ted Landau and Dan Frakes), and the "Cult of the Mac", presented by Leander Kahney (journalist for Wired News and former senior writer for MacWeek). Two well-attended tips and tricks talks: "Tricked-out X: How do Alpha MacGeeks Arrange Their OS X Workspace" and "Mac OS X Hacks" proved popular both with audience members and multiple panelists showing off their favorite tricks.

USER INTERFACE; PROGRAMMING

Developers could attend their choice of two dozen sessions on user interface design or Mac OS X programming, as well as six different programming tutorial choices on Monday. Tutorials covered AppleScript, Objective-C, Java, Perl, and Programming Cocoa (in Objective-C or Perl).

Conference sessions included "Adopting the Mac OS X User Experience in your Application", a 45-minute run-through of the Aqua Human Interface Guidelines presented by John Geleynse, Apple's user experience evangelist in Worldwide Developer Relations. This was immediately followed by a related talk, "Mac User Interface Design for New Developers" presented by Brook Conner, author of the forthcoming book "Programming Quartz: Advanced 2D Graphics on the Macintosh".

Other sessions covered Java Media, RealBasic, Objective-C, Quartz, and using AppleScript (or Perl) to automate workflow. Cocoa was presented in several sessions, ranging from "An Introduction to the Cocoa Document Architecture" to "API Techniques" to "Getting Data Onscreen with Cocoa".

SERVERS AND NETWORKING; MULTIMEDIA

Two complementary tracks included over a dozen presentations, including sessions on Open Directory/LDAP, NetInfo, Rendezvous, and the Open Source databases available for Mac OS X. Several sessions addressed the creation of web sites. Derrick Story and Rael Dornfest's session, "Building a Mac-Based Web Site", discussed secrets for QuickTime video, online iPhoto slide shows, and the best ways to make large files available to others without FTP access. This was followed by Dori Smith's "Serving Your Site from a Mac", which discussed the use of Apache and numerous additional free tools attendees could use to build a web site on Mac OS X.

Several sessions were available for users of WebObjects, including a half-day tutorial introduction to WebObjects Tools and Techniques, a 45-minute WebObjects technical overview, and a discussion of Rapid Application Development using WebObjects. Finally, for those who wanted a bit more fun, Damien Stolarz presented "How to Put up your Own TV Station on the Internet with Mac OS X".

HARDWARE

Several playful sessions focused on hardware. Dori Smith discussed how to build a Mac clone that will run Mac OS X for "a fraction of both the price and the looks" of Apple hardware; the



Fetch

Top dog.

X

Fetchsoftworks.com

Version 4.0.2 now available.

Ask yourself this question...

I use my Mac for:

- ☐ Programming as a hobby
- ☐ Exploring the depths of Mac OS X
- ☐ QuickTime Development
- ☐ Application Development
- ☐ Network Administration
- ☐ All Of The Above

...this is the answer:

 **MacTech[®]**

The Journal of Macintosh Technology and Development

**Whether you program as a hobby or are a full-time developer,
MacTech gives you the info you need from the people that know**

www.mactech.com

session was appropriately entitled "Building a Cheap, Ugly Mac". Ted Stevko provided two ways to create robots using a Macintosh in "Frankentosh: Creating Robots on the Mac".

Kent Salas shared his heavily modified G4 (with an LCD light-show panel, blue interior lights, and firewire/USB ports on the front panel; <http://www.kentsalas.com/blueiceg4>) in "Mac Mechanical Mayhem, or How to Completely Void Your Mac's Warranty". The talk was well attended and quite a few attendees stayed into the break to talk about mods with Kent.

UNIX

Perhaps unusual for a Mac OS conference (but less unusual for either O'Reilly or Mac OS X), one track was simply called Unix. Sessions in this track ranged from "Mac OS X for the Common Unix Folk" to "Migrating from Linux to Mac OS X" to "Mac OS X is Just Another Unix: Writing Portable Applications". The session entitled "From Unix to Aqua: Porting Large Unix Applications to Mac OS X" could just as easily have listed in the User Interface track as this one.

The Unix track addressed one of the newest (but steadily growing) groups of users adopting Mac OS X - the traditional Unix (Sun, Linux, BSD, etc.) users. Tim O'Reilly has commented that "Apple's 'Switch' ad campaign focuses on people making the switch from Windows, but it may be the case that there's an even larger wave of switchers from Linux and other Unix platforms." While it is the case that many UNIX users have used the Mac in the past, they have previously needed to keep two computers on their desks in order to work with both Mac OS and Unix. With Mac OS X, this is no longer the case; long-time UNIX users are trying out Mac OS X and discovering that they like it.

The panel session, "Introducing the Mac User Community to Unix Developers" (part of the "In the Large" track) was meant to bring Unix developers together with traditional Mac OS developers. The session focused on what Mac OS X developers coming from Unix backgrounds need to know about the philosophy of the Mac and the orientation of the Mac community. Panelists discussed similarities and differences between the Unix and Mac OS user communities and talked about what each group can do for the other.

CLOSING THOUGHTS

I enjoyed this conference. I've been to many technical conferences, but this was the first where everyone was like me, in that everyone used a Mac and everyone, even the presenters, used Mac OS X. I admit that WWDC is certainly Mac-oriented in its presentations, but those sessions are given by Apple. This conference was by and for developers and power users, again, people like me.

The only drawback I can mention is that I often couldn't decide which session to attend! Multiple tracks can cause that sort of difficulty. But as the week drew to a close, I was already wondering what interesting talks, tutorials, presentations, and panels will be in store next year. I'll be there. You should be sure to attend too.

REFERENCES

Tim O'Reilly's "Mac OS X Switcher Stories" article is available online at: <http://www.macdevcenter.com/pub/a/mac/2002/08/21/switch.html>.

Information on past and future O'Reilly conferences is available at: <http://conferences.oreilly.com>.

Information on O'Reilly's products and news for the Macintosh platform can be found at: <http://mac.oreilly.com>.

Use as a **first** priority, not as a last resort...



Data Rescue recovers your valuable
data before, during or after
a hard disk crisis.

PROSOFT
engineering inc.

www.prosofteng.com

1.866.428.3282



© 2002 Prosoft Engineering Inc. All rights reserved.
Data Rescue is a trademark of Prosoft Engineering Inc.
Mac and the Mac logo are trademarks of Apple Computer, Inc., in the U.S.
and other countries. The "Built for Mac OS X" graphic is a trademark
of Apple Computer, Inc., used under license.
All other trademarks are the property of their respective owners.

by Bob Boonstra, Westford, MA

PENULTIMATE

When I took over this column from Mike Scanlin back in June, 1995, I had no idea that I would still be writing it seven and one-half years later. Running the Programmer's Challenge contest has been both a rewarding and an increasingly demanding experience, but the time has come for me to move on. The title of this column, my 90th, refers, not to the name of a new Challenge, but to the fact that my next column will be the last one. Besides announcing the winner of my final Challenge, the October Area Challenge, next month will include a retrospective on the Programmer's Challenge, the problems, the contestants, the evolution of the column, and perhaps a few anecdotes. Until then, I hope you have enjoyed reading the column as much as I have enjoyed writing it.

WINNER OF THE SEPTEMBER, 2002 CHALLENGE

The August, 2000, Challenge problem asked readers to write code that would solve a sequence of chess end-game positions. Unfortunately, the problem must have been too difficult, and no entries were submitted. I thought someone might have adapted and extended the code from the problem in my first column, the June, 1995, Check Checkmate Challenge, but this was not to be. Perhaps an omen

So, taking advantage of an extended deadline for this issue, we will look at the winner of the September PhotoMosaic Challenge. Announced with a tongue-in-cheek reference to the 25th anniversary of the death (or abduction by aliens, whichever you choose to believe) of Elvis, this problem asked readers to generate a mosaic of smaller images that approximated a target image. Congratulations, once again, to **Ernst Munter** (Kanata, Ontario), for submitting the fastest and most accurate mosaic generator.

Both Ernst and second-place finisher Jan Schotsman use the Altivec programming model. Ernst divides the elements and images into "spots" of 4 pixels by 4 pixels, which allows one color plane of the spot to be represented in 64 bits – 8 bits/color for each of the 16 pixels. The image elements used to construct the mosaic are divided into "slices", or portions of the element corresponding to the desired tile size, and the slices are sorted by luminence. Ernst uses the luminence of a tile in the target image to select a position in the sorted slice array, and searches within the array to find the most appropriate slice. The depth of the search is limited to control execution time. The slice is selected to minimize the distance from the target image in RGB space. There are additional refinements to the algorithm, as described in the extensive commentary contained in the code.

I tested the entries submitted using twelve test cases, using three sets of mosaic elements, one set from lighthouse photos I

took at Isle au Haut, another set from pictures taken by my son on a trip to France, and the last set from pictures taken in the British Virgin Islands. Ernst's solution produced better mosaics in 9 of the 12 test cases, and used significant less execution time than the second-place entry. Below, I have posted an example mosaic produced by Ernst's code, along with the original image, at the MacTech web site as:

<http://www.mactech.com/progchallenge/WinningMosaic/OutputImage.jpg>
and

<http://www.mactech.com/progchallenge/WinningMosaic/InputImage.jpg>.

The table below lists, for each of the solutions submitted, the number of test cases processed correctly, the total distance between pixels of the mosaic and the target image, execution time in milliseconds, and the total score for each solution. As usual, the number in parentheses after the entrant's name is the total number of Challenge points earned in all Challenges prior to this one.

| Name | Cases Correct | Distance | Time (msecs) | Score |
|---------------------|------------------|----------|-----------------|-------|
| Ernst Munter(882) | 12 | 15598 | 92522 | 16914 |
| Jan Schotesman (18) | 12 | 16754 | 337847 | 22198 |
| Tony Cooper (20) | 12 | 18625 | 440358 | 25951 |

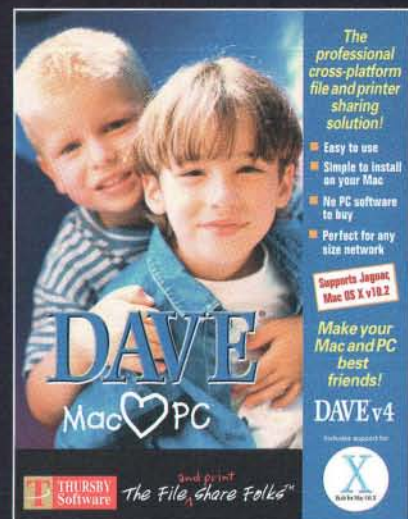
TOP CONTESTANTS ...

Listed here are the Top Contestants for the Programmer's Challenge, including everyone who has accumulated 20 or more points during the past two years. The numbers below include points awarded over the 24 most recent contests, including points earned by this month's entrants.

| Rank | Name | Points (24 mo) | Wins (24 mo) | Total Points |
|------|-------------------|-------------------|-----------------|-----------------|
| 1. | Munter, Ernst | 241 | 8 | 902 |
| 2. | Saxton, Tom | 65 | 2 | 230 |
| 3. | Taylor, Jonathan | 54 | 2 | 90 |
| 4. | Stenger, Allen | 53 | 1 | 118 |
| 5. | Hart, Alan | 34 | 1 | 59 |
| 6. | Cooper, Tony | 27 | 1 | 27 |
| 7. | Rieken, Willeke | 22 | 1 | 134 |
| 8. | Sadetsky, Gregory | 22 | 0 | 24 |
| 9. | Landsbert, Robin | 22 | 1 | 22 |
| 10. | Schotsman, Jan | 21 | 0 | 28 |
| 11. | Gregg, Xan | 20 | 1 | 140 |
| 12. | Mallett, Jeff | 20 | 1 | 114 |
| 13. | Wihlborg, Claes | 20 | 1 | 49 |
| 14. | Truskier, Peter | 20 | 1 | 20 |

DAVE[®] v4

NOW, MORE FEATURES FOR THE SERIOUS MAC USER!



“DAVE SAVED MY SANITY”

IT Analyst Timothy Hixon, Los Rios Community College System explained, “We didn’t have time to learn another network operating system. The Macs needed to fit seamlessly and efficiently into our district’s computer network, which used Windows NT file servers. DAVE was the answer.”

Compare!

- Shares files and printers between Macs and PCs
- Supports Microsoft standard NTFS file format
- Provides three security options
- Offers fully integrated and efficient browsing
- Insures compatibility with Services for Macintosh
- Provides NT Domain Login

MacAddict
RATED



Macworld



“DAVE not only supports Macintosh computers still running OS 9, it enhances Jaguar’s SMB capabilities for the advanced and professional user”

SHARE FILES AND PRINTERS - Mac/PC, PC/Mac

“DAVE is, without any doubt, the best solution for integrating your Mac into a PC environment.”

-Univers Macworld



www.thursby.com

T THURSBY
Software

and print
The File, Share Folks™

Here is Ernst winning PhotoMosaic solution.

MOSAIC-V2.CP

Copyright © 2002
Ernst Munter

```
/*  
"Photo Mosaic"  
Version 2
```

The task is to assemble a photomosaic resembling a "desired image" from tiles cut out from a number of "element images". The criteria are closeness of the fit in terms of the color distance (RMS sum of the RGB color components of all pixels) between the desired image and the mosaic image, and speed.

Solution Strategy

There is a trade-off between achievable color distance and speed. The number of degrees of freedom given - choice of tile size above a minimum, position of the tile cut from the chosen element - is too large to allow for an exhaustive search of all possible tile cuts ("element slices") for all possible tile sizes.

I select the two smallest tile sizes that can be used to fill the mosaic. Each element, as well as the tiles to be matched, are represented as an array of "Spots", a spot being a 4 by 4 cluster of pixels.

In a next step, all possible slices of the elements are assigned a luminance value, and the slice is stored in an array of lists, the array being indexed by luminance.

To match a desired image then, each tile of the mosaic is processed independently. The luminance of the tile (in the desired picture) is used as an index into the slice array. The position in the slice array will be in the center of a range of slices of similar luminance as the tile to be matched.

In the next step, all slices within the indicated range are compared (RMS color distance) with the tile, and the closest slice identified.

Up to this point, only slices on the 4-pixel grid were considered. The closest identified slice is then taken as the center of a small area within the element, and slices on a 1-pixel grid are evaluated to make the final selection.

The method to control running time is to choose the size of the range of slices within the slice array that are evaluated.

Since the running time penalty 1% per second is fixed, but running time is roughly proportional to image size, the range is chosen as the reciprocal of the image size times an arbitrary factor, to yield about a 10 to 20% penalty on a large (2-3Mpixel) image.

Vector Processor

The processor in the G4 Macs contains a vector processing unit (AltiVec). This processor is very efficient at processing up to 16 bytes in parallel, just the thing to process pixels in parallel, as long as the pixels are represented in planar form. For example 3 vectors can hold 16 pixels (16 bytes of the same color in each vector). An RGB representation (32-bit pixels) of a Spot, containing 4 by 4 pixels of an image, can be efficiently converted into a planar vector representation.

This then allows the RMS color difference of sets of 16 pixels to be computed with a very small number of instructions.

I use an approximation of the true RMS value, by adding the largest absolute differences between corresponding pixel planes (R, G, or B) to 5/16 of the sum of the absolute differences of the other planes.

This is the most busy function of the algorithm; it takes 25 vector instructions to compute the sum of the RMS color difference of 16 pixel pairs.

Version 2 changes are described in "MosaicClasses-v2.h".

```
*/  
#include "Mosaic.h"  
#include "MosaicClasses-v2.h"
```

```
static Ernst::Mosaic* gM;// Mosaic classes are in namespace Ernst
```

InitMosaic

```
void InitMosaic(  
    short numMosaics,  
    /* number of pixmaps from which the mosaic should be created */  
    const PixMapHandle element[])  
    /* element[i] is a PixMapHandle to the i-th image used in constructing mosaic */  
)
```

```
// Called once to initialize the elements in a new Mosaic class.  
// Mosaic element images are pixel-locked in InitMosaic, and remain locked  
// until TermMosaic() is called.  
{  
    assert(gM==0);  
    gM=new Ernst::Mosaic(element,numMosaics);  
}
```

Mosaic

```
void Mosaic(  
    const PixMapHandle desiredImage,  
    /* PixMapHandle populated with the desired image to be constructed */  
    const Rect minPieceSize,  
    /* mosaic pieces must be of this size or larger */  
    PixMapHandle mosaic,  
    /* PixMapHandle to preallocated image in which the mosaic is to be placed */  
    /* initialized to black */  
    MosaicPiece *piece,  
    /* pointer to array of mosaic pieces */  
    /* populated by your code */  
    long *numMosaicPieces  
    /* number of mosaic pieces created by your code */  
)  
// May be called multiple times with different desiredImages,  
// and possibly with a different minPieceSize.  
// The function locks/unlocks the image pixels,  
// The mosaic is prepared (a private set of image parameters of the desired image),  
// solved (element slices assigned to mosaic tile coordinates)  
// and cleaned up (the private set deleted).  
{  
    LockPixels(desiredImage);  
    LockPixels(mosaic);  
  
    *numMosaicPieces =  
        gM->PrepareMosaic(desiredImage,minPieceSize);  
  
    gM->SolveMosaic(mosaic,piece);  
  
    gM->Cleanup();  
  
    UnlockPixels(mosaic);  
    UnlockPixels(desiredImage);  
}
```

TermMosaic

```
void TermMosaic(void)  
    /* deallocate any memory allocated by InitMosaic or multiple Mosaic calls */  
{  
    delete gM;  
    gM=0;  
}
```

MOSAICCLASSES-V2.H

```
#ifndef MOSAIC_CLASSES_H  
#define MOSAIC_CLASSES_H
```

```
/*  
"Photo Mosaic"
```

Version 2:

Simplification in luminance retrieval:

cast return value directly from memory to unsigned long.

In MatchTile():

Try the best slice from the previous tile first.

Alternative scheme (SCHEME = 2) added:

Do preliminary match on the average color of each spot, instead of on the individual pixels of the spot.

This provides a significant speedup, at the expense of slightly higher overall distances, resulting in similar scores.

The original scheme 1 is preferred because it produces a slightly better match (kSearchRangeFactor set to optimize for a 1% / sec time penalty).

```
*/  
#define NDEBUG  
#include <cassert>  
#include <cstring>
```

```
#define SCHEME 1
```

```
// A namespace is created to avoid name clashes with Apple headers or test code  
namespace Ernst {
```

```
typedef unsigned char uchar;  
typedef unsigned short ushort;  
typedef unsigned long ulong;  
typedef ulong Pixel;
```

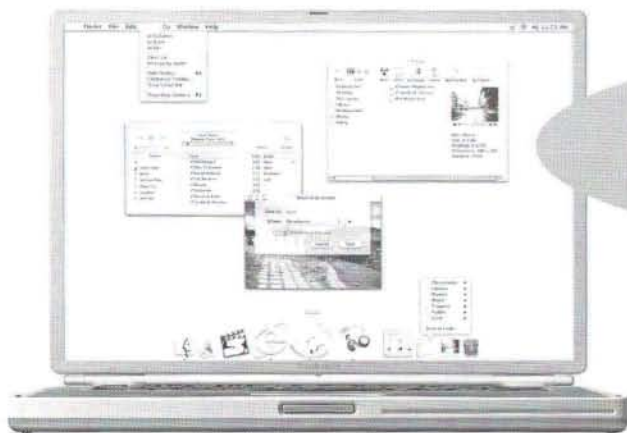

Special Small Dogs

Here are just a few of the photos our customers have sent us of their special dog friends!



Check out all the other special Small Dog photos and send us one of your own at www.smalldog.com!

Small Dog's Special



PowerBook G4/667 Titanium
\$2499

- 512MB RAM
- 30gb Hard Drive
- Combo Drive
- Airport card and base station



Purchase from an Apple Specialist!

Small Dog Electronics is proud to have earned Apple's prestigious designation of Apple Specialist. Small Dog Electronics has exceeded all of Apple's stringent requirements for its Specialists and has taken it one step further.

Small Dog sells only the most powerful personal computers in the world—every single one of them an Apple Macintosh! In addition to the rigorous Apple training program, each Small Dog employee has chosen a specific area of expertise to concentrate upon. You can be assured that whomever you talk to at Small Dog, from our sales engineers to our shipping department, you are talking to a trained, enthusiastic, Apple Macintosh Specialist!

Small Dog Electronics is also an Authorized Apple Service Provider Plus. We have certified Apple Technicians that utilize genuine Apple parts for all repairs and provide technical support for our customers based upon their experience in upgrading and supporting thousands of Apple Macintosh computers for our customers!

Talk To an Apple Professional!

Did you know that when you call Small Dog Electronics, you are most likely talking to an Apple Product Professional? Each year, we participate in Apple's on-line courses and seminar training programs to learn as much as we can about the products that we sell and service!

Exclusive Top Dog Membership Treats

For each dollar purchased on-line at the Small Dog web site, you will receive a doggie treat. You can redeem your accumulated treats for Small Dog or Apple apparel and other products. You are automatically enrolled and begin accumulating treats in the Top Dog Club with your first purchase. Remember the more you buy, the more treats for you!



**Small Dog
Electronics**

1673 Main Street
Waitsfield, VT 05673 USA
Phone: 802-496-7171
Online: smalldog.com



Apple Specialist


```

typedef vector unsigned char vuchar;
typedef vector unsigned short vushort;
typedef vector signed long vlong;
typedef vector unsigned long vulong;
typedef vector bool char vboolchar;

// Macros extract the red, green, and blue components of a 32-bit RGB pixel
#define mRED(pixel)      (0xFF & ((pixel)>>16))
#define mGREEN(pixel)   (0xFF & ((pixel)>>8))
#define mBLUE(pixel)    (0xFF & (pixel))

enum {
    kSpotSize = 4, // 4 by 4 pixels
    kSpotHeight = kSpotSize,
    kSpotWidth = kSpotSize,
    kNumLuminanceLevels = 1+255*3, // sum of R+G+B ranges from 0 to 755
    kSearchRangeFactor = 1000*1000*1000 // chosen to yield reasonable
                                         // run times
};

// Vector function, computes the absolute difference between all pairs of
// the vector-elements in a and b (T is an unsigned vector type)
template <class T> inline T AbsDifference(T a,T b)
{
    T max=vec_max(a,b);
    T min=vec_min(a,b);
    return vec_sub(max,min);
}

// Vector functions, computes an estimate of the RMS sum of the color
// differences of all pairs of color planes 1 (vr1,vgl,vb1) and 2 (vr2,vg2,vb2)
// The algorithm is
//   max = the largest among the components R, G, and B
//   minA and minB are the other two components
//   RMS ~ max + (minA+minB)*5/16
// Returns the sum of the sixteen RMS distances so computed
inline void VColorDistance(
    vuchar vr1,
    vuchar vgl,
    vuchar vb1,
    vuchar vr2,
    vuchar vg2,
    vuchar vb2,
    long* result)
{
    vuchar dRed=AbsDifference<vuchar>(vr1,vr2);
    vuchar dGreen=AbsDifference<vuchar>(vgl,vg2);
    vuchar dBlue=AbsDifference<vuchar>(vb1,vb2);

    vuchar vMax=vec_max(dRed,dGreen);
    vuchar vMinA=vec_min(dRed,dGreen);
    vuchar vMinB=vec_min(vMax,dBlue);
    vMax=vec_max(vMax,dBlue);

    vulong k4=(vulong)(4,4,4,4);
    vuchar k5=(vuchar)(5,5,5,5, 5,5,5,5, 5,5,5,5, 5,5,5,5);

    vulong k0=(vulong)(0,0,0,0);

    vulong partSumMax=vec_sum4s(vMax,k0);
    vlong sumMax=vec_sums((vlong)partSumMax,(vlong)k0);

    vulong intermediateMinA=vec_msum(vMinA,k5,k0);//*5
    vulong intermediateMinB=vec_msum(vMinB,k5,k0);//*5
    intermediateMinA=vec_add(intermediateMinA,intermediateMinB);

    intermediateMinA=vec_sr(intermediateMinA,k4);///16

    vlong sum=vec_sums((vlong)intermediateMinA,sumMax);
    sum=vec_splat(sum,3);

    vec_ste(sum,0,result);
}

inline void VColorDistanceLong(vulong v1,vulong v2,long* result)
{
    vulong delta=AbsDifference<vulong>(v1,v2);
    vulong dRed=vec_splat(delta,1);
    vulong dGreen=vec_splat(delta,2);
    vulong dBlue=vec_splat(delta,3);

    vulong vMax=vec_max(dRed,dGreen);
    vulong vMinA=vec_min(dRed,dGreen);
    vulong vMinB=vec_min(vMax,dBlue);

```

```

    vMax=vec_max(vMax,dBlue);

    vulong k4=(vulong)(4,4,4,4);
    vushort k5=(vushort)(5,5,5,5, 5,5,5,5);

    vulong k0=(vulong)(0,0,0,0);

    vulong sum=vec_add(vMinA,vMinB);
    sum=vec_msum((vushort)sum,k5,k0); // *5
    sum=vec_sr(sum,k4); // /16
    sum=vec_add(sum,vMax);

    vec_ste((vlong)sum,0,result);
}

class Spot
{
    //A Spot represents a square of 4 by 4 pixels,
    //This is the smallest screen element representable
    //as a set of vectors, each vector in a different color plane
    //Color planes are R, G, B, and alpha,
    //The alpha values (x) are ignored.
    //
    //
    class Spot
    {
        vuchar xrgb[kSpotSize];
    public:
        void Init(Pixel* pixels,int rowPitch)
        //Initializes a full spot of 16 pixels and computes spot luminance.
        //We copy the 16 pixels into vectors,
        //using memcpy because the original pixels may not be 16-byte aligned
        {
            for (int i=0;i<kSpotSize;i++,pixels+=rowPitch)
                std::memcpy(xrgb+i,pixels,sizeof(vuchar));

            MakeColorPlanes();
        }

        vlong ColorDistance(Spot& S)
        //Wrapper for the inline vector function VColorDistance()
        {
            long result;
            VColorDistance(
                xrgb[1],xrgb[2],xrgb[3],
                S.xrgb[1],S.xrgb[2],S.xrgb[3],
                &result);
            return result;
        }

        vlong ColorDistanceLong(Spot& S)
        //Wrapper for the inline vector function VColorDistanceLong()
        {
            long result;
            VColorDistanceLong(
                (vulong)xrgb[0]),
                (vulong)(S.xrgb[0]),
                &result);
            return result;
        }

        vlong Luminance() const
        //Returns the spot luminance stored at xrgb[0]
        {
            return *((ulong*)(xrgb));
        }

    private:
        void MakeColorPlanes()
        //Transforms 4 rows of 4 pixels each into 3 color planes and a luminance plane.
        //Uses vector operations to efficiently move the 16 R,G,B pixel values around.
        {
            vuchar vXRGB0=xrgb[0];
            vuchar vXRGB1=xrgb[1];
            vuchar vXRGB2=xrgb[2];
            vuchar vXRGB3=xrgb[3];

            vuchar vRB01=vec_pack((vushort)vXRGB0,(vushort)vXRGB1);
            //extract RB
            vuchar vRB23=vec_pack((vushort)vXRGB2,(vushort)vXRGB3);

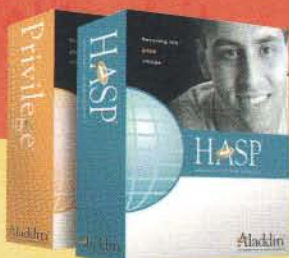
            vuchar k8=(vuchar)(8,8,8,8,8,8,8,8,8,8,8,8,8,8,8,8);
            vuchar v0XRG0=vec_sro(vXRGB0,k8);
            //shift right

```




CHICKENS SHOULD ROAM FREE. YOUR SOFTWARE SHOULDN'T.

Try our
Piracy Calculator.
See how much revenue may
have already flown the
coop at hasp.com/cal.



HASP® & Privilege®

Don't let "free roaming" software result in lost revenues — take advantage of the superior software licensing solutions offered by Aladdin Knowledge Systems. Did you know some businesses lose almost 50% of their revenue due to software piracy? That's why over 25,000 companies trust their software

licensing to Aladdin. Whether you prefer the proven reliability of HASP® dongles (USB or parallel port) or the state-of-the-art downloading of Privilege™, Aladdin products have a 99.97% success rate. Call us at 1-800-562-2543 to prevent your profits from flying the coop.

Aladdin®
SECURING THE GLOBAL VILLAGE
eAladdin.com


```

vuchar v0XRG1=vec_sro(vXRGB1,k8);
vuchar v0XRG2=vec_sro(vXRGB2,k8);
vuchar v0XRG3=vec_sro(vXRGB3,k8);

vuchar vXG01=vec_pack((vushort)v0XRG0,(vushort)v0XRG1);
vuchar vXG23=vec_pack((vushort)v0XRG2,(vushort)v0XRG3);
// extract XG

vuchar vB=vec_pack((vushort)vRB01,(vushort)vRB23);
// extract B

xrgb[3]=vB; // store B
vuchar vG=vec_pack((vushort)vXG01,(vushort)vXG23);
// extract G

xrgb[2]=vG; // store G

vuchar v0R01=vec_sro(vRB01,k8); //shift right
vuchar v0R23=vec_sro(vRB23,k8);

vuchar vR=vec_pack((vushort)v0R01,(vushort)v0R23);
// extract R

xrgb[1]=vR; // store R

// Add all R G B components to obtain the spot luminance:
vulong k0=(vulong)(0,0,0,0);
vulong sumB = vec_sum4s(vB,k0);
vulong sumG = vec_sum4s(vG,k0);
vulong sumR = vec_sum4s(vR,k0);
vulong sumBG= vec_add(sumB,sumG);
vulong sumRBG=vec_add(sumBG,sumR);
vlong lrgb = vec_sums((vlong)sumRBG,(vlong)k0);
#if SCHEME==2
// Compute the individual RGB sums (= average color of a spot)
// lrgb = luminance, red, green, blue
lrgb = vec_sld((vulong)lrgb,k0,12);
sumR = (vulong)vec_sums((vlong)sumR,(vlong)k0);
sumR = vec_sld(sumR,k0,8);
sumG = (vulong)vec_sums((vlong)sumG,(vlong)k0);
sumG = vec_sld(sumG,k0,4);
sumB = (vulong)vec_sums((vlong)sumB,(vlong)k0);
lrgb = vec_or(lrgb,sumR);
lrgb = vec_or(lrgb,sumB);
lrgb = vec_or(lrgb,sumG);
#else
// Splat luminance into all 4 words
// lrgb = luminance only
lrgb = vec_splat(lrgb,3);
#endif
xrgb[0]=(vuchar)lrgb; // store luminance (and avg RGB in SCHEME 2)
};

struct Slice
//
// A Slice is an element in a singly linked list.
// A slice identifies a slice (of arbitrary dimension) in an element
// by its top left corner.
//
//
//
struct Slice
{
    Slice* next;
    short elementId;
    uchar top;
    uchar left;

    Slice():next(0),elementId(0),top(0),left(0) {}
    Slice(Slice* s,long id,long t,long l):
        next(s),
        elementId(id),
        top(t),
        left(l)
    {}
};
typedef Slice* SlicePtr;

class Image
//
// An Image is represented by a reference to the original (32-bit) RGB pixels
// The pixels may be converted into a representation as spots.
// All elements, the desired image, the mosaic, and the current tile, are

```

```

// represented by an image structure.
//
// When used for elements and the current tile, spots are allocated (planar copies
// of the pixels), and corresponding data members initialized and used.
//
//
class Image
{
    Pixel* pixels; // pixel pointer to original pixels
    Spot* spots; // spots are allocated for elements and tiles
    bool allocatedPixels;
    bool allocatedSpots;

    short width; // width of the image in pixels
    short height;
    short rowPitch; // rowBytes / 4
    long elementId; // needed when element is copied to a mosaic piece

    long numSpots; // spots related dimensions
    short numSpotsH; // horizontal
    short numSpotsV; // vertical

public:
// Image Constructors
    Image():
        pixels(0),spots(0),
        allocatedPixels(false),allocatedSpots(false)
    {}
    Image(Pixel* c_pixels,
        Spot* c_spots,
        bool c_allocPixels,
        bool c_allocSpots,
        long c_Width,
        long c_Height,
        long c_RowPitch,
        long id):
        pixels(c_pixels),
        spots(c_spots),
        allocatedPixels(c_allocPixels),
        allocatedSpots(c_allocSpots),
        width(c_Width),
        height(c_Height),
        rowPitch(c_RowPitch),
        elementId(id)
    {}

// Image Destructor
    ~Image()
    {
        if (allocatedSpots && spots) delete [] spots;
        if (allocatedPixels && pixels) delete [] pixels;
    }

// Image accessor functions
    long Width() const {return width;}
    long Height() const {return height;}
    ulong* Pixels(int row,int col) const
    {
        return pixels+col+row*rowPitch;
    }
    long RowPitch() const {return rowPitch;}

    void Init(const PixMapHandle pm,long id)
// Initializes an Image from a QuickDraw PixMapHandle
    {
        pixels=(Pixel*)GetPixBaseAddr(pm);
        allocatedPixels=false;
        allocatedSpots=false;
        width=(**pm).bounds.right-(**pm).bounds.left;
        height=(**pm).bounds.bottom-(**pm).bounds.top;
        rowPitch=(0x3FFF & (**pm).rowBytes)/sizeof(Pixel);
        elementId=id;
        spots=0;
    }

// Update Functions change the image information for a previously initialized image.
// Saves deleting/reconstructing successive slices or tiles during the search.
    void Update(Pixel* c_pixels,long c_Width,long c_Height)
    {
        pixels=c_pixels;
        width=c_Width;
        height=c_Height;
    }
};

```



```

void Update(Pixel* c_pixels)
{
    pixels=c_pixels;
}

void AllocateSpots()
// Computes number of spots, and allocates new spots for the image.
{
    numSpotsH=width/kSpotWidth;
    numSpotsV=height/kSpotHeight;
    numSpots=numSpotsH*numSpotsV;
    spots = new Spot[numSpots];
    allocatedSpots=true;
}

Spot* InitSpots()
// Allocates spots if necessary.
// Initializes the spots on a 4-pixel grid (kSpotWidth=kSpotHeight=4).
// If image width or height are not multiples of four, pixels
// along the bottom and right edges are not represented in spots.
// Returns the (possibly newly allocated) spots.
{
    if (!spots) AllocateSpots();

    numSpotsH=width/kSpotWidth;
    numSpotsH=height/kSpotHeight;

    Spot* SP=spots-1;
    Pixel* pRow=pixels;
    Pixel* pCol=pRow;
    long col;
    for (int row=0;row<numSpotsV;row++)
    {
        for (col=0;col<numSpotsH;col++,pCol+=kSpotWidth)
        {
            (++SP)->Init(pCol,rowPitch);
        }

        pRow+=kSpotHeight*rowPitch;
        pCol=pRow;
    }
    return spots;
}

long PrepareSlices(Slice** sliceIndex,long tileRangeH,
                  long tileRangeV)
// For each possible slice in an element, computes slice luminance and
// attaches the slice to a list in sliceIndex[luminance].
// Returns the number of slices.
{
    assert(spots);
    long spotRangeH=numSpotsH-tileRangeH;
    long spotRangeV=numSpotsV-tileRangeV;
    for (long spotRow=0;spotRow<spotRangeV;spotRow++)
    {
        for (long spotCol=0;spotCol<spotRangeH;spotCol++)
        {
            ulong luminance=
                SliceLuminance(spotRow,spotCol,tileRangeH,tileRangeV);
            assert(luminance < kNumLuminanceLevels);
            Slice* newSlice=new
                Slice(sliceIndex[luminance],elementId,
                    4*spotRow,4*spotCol);
            sliceIndex[luminance]=newSlice;
        }
    }
    return spotRangeV*spotRangeH;
}

long SliceLuminance(long spotRow,long spotCol,
                  long tileRangeH,long tileRangeV)
// Returns a normalized sum of luminance values of a tile or slice.
{
    Spot* S=spots+spotCol+spotRow*numSpotsH;
    ulong sum=0;
    for (long r=0;r<tileRangeV;r++)
    {
        for (long c=0;c<tileRangeH;c++)
        {
            ulong lu=S[c].Luminance();
            sum+=lu;
        }
        S+=numSpotsH;
    }
}

```

```

    sum /= tileRangeV*tileRangeH*kSpotSize*kSpotSize;
    assert(sum<kNumLuminanceLevels);
    return sum;
}

long TileLuminance()
// Returns a normalized sum of luminance values of a tile.
{
    long lum=SliceLuminance(0,0,numSpotsH,numSpotsV);
    return lum;
}

ulong RawDistance(Slice* slice,Image& tile,ulong minDistance)
// Computes the raw color distance between a slice in an element, and the image tile.
// Breaks computation off early if (previously found) minimum distance is exceeded.
// Returns the accumulated color distance.
{
    long bottom=slice->top/4+tile.numSpotsV;
    long right=slice->left/4+tile.numSpotsH;
    Spot* elementSpot=spots+slice->top/4*numSpotsH;
    Spot* tileSpot=tile.spots;
    ulong diff=0;
    for (long row=slice->top/4;row<bottom;row++)
    {
        for (long col=slice->left/4;col<right;col++)
        {
            #if SCHEME==2
                long d=elementSpot[col].ColorDistanceLong(tileSpot[0]);
            #else
                long d=elementSpot[col].ColorDistance(tileSpot[0]);
            #endif
            diff+=d;
            if (diff>minDistance)
                return diff;
            tileSpot++;
        }
        elementSpot+=numSpotsH;
    }
    return diff;
}

ulong ColorDistance(Image& tile)

```

high quality - competitive rates - 16 years experience - award winning

Full Spectrum Software Development & Testing

Device Drivers
Porting Plug-ins
TCP/IP
Carbon / OSX

Cross Platform Development

One Bridge Street
Newton, MA 02458

617-965-0029

www.FullSpectrumSoftware.com

reliable - high quality - competitive rates - efficient - award winning - qa services - reputable

reliable - high quality - competitive rates - efficient - award winning - qa services - reputable

competitive rates - 16 years experience - award winning - high quality


```
//The same purpose as Image::RawDistance, but used when the class instance of
// the image is a custom copy of a slice (on a 1-pixel grid).
//This avoids having to deal with unaligned spots.
```

```
{
    assert(spots);
    assert(tile.spots);
    assert(numSpots==tile.numSpots);
    ulong diff=0;
    for (int i=0;i<numSpots;i++)
    {
        long d=spots[i].ColorDistance(tile.spots[i]);
        diff+=d;
    }
    return diff;
}

ulong MatchVicinity(Slice& slice,Image& tile)
//After having found a good candidate slice,this function is used to micro-align
// the slice on a 1-pixel grid,in order to obtain a possibly better match.
//A +3 pixel vicinity of the original slice position is explored.
{
    enum {DELTA=3};
    ulong minDistance=0xFFFFFFFF;
    int r0=slice.top-DELTA;if (r0<0) r0=0;
    int c0=slice.left-DELTA;if (c0<0) c0=0;
    int r1=slice.top+DELTA;
    if (r1>=height-tile.height) r1=height-tile.height;
    int c1=slice.left+DELTA;
    if (c1>=width-tile.width) c1=width-tile.width;
    Image sliceImage
    (Pixels(r0,c0),0,0,0,tile.width,tile.height,rowPitch,0);
    for (int row=r0;row<r1;row++)
    {
        for (int col=c0;col<c1;col++)
        {
            sliceImage.Update(Pixels(row,col));
            sliceImage.InitSpots();
            ulong colorDistance=sliceImage.ColorDistance(tile);
            if (colorDistance<minDistance)
            {
                minDistance=colorDistance;
                slice.top=row;
                slice.left=col;
            }
        }
    }
    return minDistance;
}

void CopyToPiece(
    MosaicPiece& piece,int top,int left,
    int pieceWidth,int pieceHeight,
    int sliceTop,int sliceLeft)
// Copies the pixels identified by a slice in an element, to a MosaicPiece.
{
    piece.elementIndex=elementId;
    Rect elementRect=(sliceTop,sliceLeft,sliceTop+pieceHeight,
        sliceLeft+pieceWidth);
    piece.elementRect=elementRect;
    Rect mosaicRect={top,left,top+pieceHeight,left+pieceWidth};
    piece.mosaicRect=mosaicRect;
}

void CopySlice(Image& E,
    int top,int left,
    int pieceWidth,int pieceHeight,long sliceTop,long sliceLeft)
// Copies the pixels identified by a slice in an element, into the mosaic image
{
    int srcRow=sliceTop,srcCol=sliceLeft; //TL corner of slice
    Pixel* src=E.Pixels(srcRow,srcCol); // slice pixels
    Pixel* dest=pixels+left+top*rowPitch; // mosaic pixels
    for (int row=0;row<pieceHeight;row++)
    {
        for (int col=0;col<pieceWidth;col++)
        {
            dest[col]=src[col];
        }
        src+=E.rowPitch;
        dest+=rowPitch;
    }
};
```

class Mosaic

```
////////////////////////////////////
//
//The class Mosaic is initialized by InitMosaic() and destroyed in TermMosaic().
// It provides the links to the elements, builds tiles as required,
// and matches the element slices to the desired image(s).
//
////////////////////////////////////
class Mosaic
{
//The following data members are defined when InitMosaic() function runs
const PixMapHandle* element;
long numElements;
Image* elementImages;

//The following data members vary with the image and are defined when Mosaic()
// function runs
Slice** sliceIndex; //Array of linked lists of slices

Image* desiredImage; // the desired image to be matched
short imageWidth;
short imageHeight;

short tileWidth; // min tile width compatible with desired image
short tileHeight; // min tile height compatible with desired image
short tileRangeH; // tile width in terms of spots
short tileRangeV; // tile height in terms of spots
long numTileColsNarrow; // number of narrow columns of tiles of tileWidth
long numTileCols; // total number of tile columns (narrow + wide)
long numTileRowsShort; // number of short rows of tiles of tileHeight
long numTileRows; // total number of tile rows (short + tall)
long numTiles; // = number of mosaic pieces
long searchRange; // value controlling extent of search in sliceIndex
short previousTileRangeH; // if successive calls to Mosaic() result in identical
short previousTileRangeV; // tileRanges, we can avoid some recomputations.

public:
//An instance of class Mosaic is constructed with the element images to be used later.
Mosaic(const PixMapHandle c_element[],long c_numElements) :
    element(c_element),
    numElements(c_numElements),
    elementImages(new Image[c_numElements]),
    sliceIndex(0),
    desiredImage(0),
    searchRange(0),
    previousTileRangeH(0),
    previousTileRangeV(0)
{
    for (int i=0;i<numElements;i++)
    {
        LockPixels(element[i]); // element pixels locked in constructor
        elementImages[i].Init(element[i],i);
        elementImages[i].InitSpots();
    }
}

// Destructor runs when TermMosaic is called
~Mosaic()
{
    for (int i=0;i<numElements;i++)
    {
        UnlockPixels(element[i]); // element pixels unlocked in destructor
    }
    if (desiredImage) delete desiredImage;
    DeleteSliceIndex();
    if (elementImages) delete [] elementImages;
}

long PrepareMosaic(const PixMapHandle c_desiredImage,
    const Rect c_minPieceSize)
// Called from Mosaic(), to prepare the image and the elements for the search.
// Chooses a (initial) search range, to achieve reasonable quality and run time.
//The plan was to update this number periodically as the search progresses (TBD).
// Returns the number of tiles that will be used to construct the mosaic.
{
    PrepareDesiredImage(c_desiredImage,c_minPieceSize);
    PrepareElements();
    ChooseInitialSearchRange();
    return numTiles;
}

double SolveMosaic(PixMapHandle mosaic,MosaicPiece *piece)
```


Your Data Isn't an Important Part of the Job — It *IS* the Job.

VXA® FireWire

The High-Performance Tape Storage Solution For Apple Users

*"I can stick
my entire
hard drive
onto a tape
7 times
—that's just
plain cool!"*

Other backup media, such as CD-RWs and DVD-RAMs, simply can't compare to VXA-I tape technology when it comes to capacity, transfer rate and overall price.

Fast, economical and easy-to-use, VXA FireWire offers:

Ultimate File Security

- 100% Data Restore and Interchange
- Plug-and-Play Connectivity

Sharable Media

- Multi-Gigabyte File Transport
- Portable, Hot-Pluggable

Real Time Digital Video

Cross-Platform Compatibility

- FireWire/IEEE 1394/iLink Supported By All Major Manufacturers



| Technology | Capacity in MBs | Transfer Rate in MB/sec | Price per MB |
|-----------------|-----------------|-------------------------|---------------|
| VXA Tape | 33000 | 3 | \$0.03 |
| Imation Travan | 10000 | 1 | \$0.05 |
| DVD RAM | 9400 | 2.7 | \$0.05 |
| CD RW | 700 | 1.9 | \$0.50 |
| Zip | 250 | 1.2 | \$0.76 |

LOWEST
COST per MB
OF ANY
Technology

Call Exabyte sales at 1-800-774-7172
or visit us on the web at www.exabyte.com



Tape Storage By
Exabyte®


```

// Each tile of the mosaic is independently chosen, after comparison with the
// selected range of candidate slices.
{
    Pixel* P=desiredImage->Pixels(0,0);
    long rowPitch=desiredImage->RowPitch();
    Pixel* mosaicP=(Pixel*)GetPixBaseAddr(mosaic);
    assert(mosaicP);
    long mosaicRowPitch=
        (0x3FFF & (**mosaic).rowBytes)/sizeof(Pixel);

    assert(mosaicRowPitch==desiredImage->RowPitch());
    Image mosaicImage;
    mosaicImage.Init(mosaic,0);
    long pixelRow=0;
    long pieceHeight;
    // preallocate largest tile
    Image tileToMatch(0,0,0,0,tileWidth+1,tileHeight+1,
        rowPitch,1001);
    tileToMatch.AllocateSpots();

    Slice bestSlice;
    SlicePtr sliceP=0;

    for (long tileRow=0; tileRow<numTileRows;
        tileRow++,pixelRow+=pieceHeight)
    {
        long pixelCol=0;
        pieceHeight=(tileRow<numTileRowsShort)?tileHeight:tileHeight+1;

        long pieceWidth;
        for (long tileCol=0; tileCol<numTileCols;
            tileCol++,pixelCol+=pieceWidth)
        {
            pieceWidth=
                (tileCol<numTileColsNarrow)?tileWidth:tileWidth+1;
            tileToMatch.Update(P+pixelCol+pixelRow*rowPitch,pieceWidth,
                pieceHeight);
            tileToMatch.InitSpots();

            //The call to MatchTile, once per mosaic piece, obtains the best matching slice on
            // the 4-pixel grid.
            MatchTile(tileToMatch,&sliceP);
            assert(sliceP);

            // Starting from the slice on the 4-pixel grid, MatchVicinity scans the neighborhood
            // of this slice in an effort to obtain the "best slice".
            bestSlice=*sliceP;

            elementImages[bestSlice.elementId].MatchVicinity(bestSlice,
                tileToMatch);

            //The best slice is copied into the next MosaicPiece structure,
            elementImages[bestSlice.elementId].CopyToPiece(
                *piece++,pixelRow,pixelCol,
                pieceWidth,pieceHeight,
                bestSlice.top,bestSlice.left);

            // ...and copied into the mosaic.
            mosaicImage.CopySlice(
                elementImages[bestSlice.elementId],
                pixelRow,pixelCol,pieceWidth,pieceHeight,
                bestSlice.top,bestSlice.left);
        }
    }
    return 0;
}

void Cleanup()
// Cleanup is needed after each desired image has been made into a mosaic, to delete it.
{
    delete desiredImage;
    desiredImage=0;
}

private:
void DeleteSliceIndex()
// DeleteSliceIndex contains a loop to delete all individually allocated slices.
{
    if (!sliceIndex) return;
    for (int i=0;i<kNumLuminanceLevels;i++)
    {
        Slice* S=sliceIndex[i];
        while (S)

```

```

{
    Slice* nextSlice=S->next;
    delete S;
    S=nextSlice;
}
delete [] sliceIndex;
}

long FindTileSize(long x,long t,long& n,long& m)
// Solves the diophantine equation: n*t + m*(t+1) == x.
// x = image size (width or height),
// t = initially the minimum tile size,
// sets n (number of tiles size t) and m (number of tiles size t+1),
// This function is useful in finding the smallest possible tile dimensions,
// that are compatible with the image dimensions.
// The assumption is that smaller tiles provide better matches
// Returns t which may have been increased from the initial value.
{
    for (;;)
    {
        n=x/t;
        m=x-n*t;
        n=(x-m*(t+1))/t;
        if (n>=0)
            break;
        t++;
    }
    return t;
}

void PrepareDesiredImage(const PixMapHandle c_desiredImage,
    const Rect c_minPieceSize)
// Analyses the dimensions of the desired image, and minPieceSize,
// and determines tiling constants.
// Determines the total number of mosaic pieces needed (numTiles).
{
    desiredImage=new Image;
    desiredImage->Init(c_desiredImage,-1);

    const Rect& iBounds=(**c_desiredImage).bounds;
    imageWidth=iBounds.right-iBounds.left;
    imageHeight=iBounds.bottom-iBounds.top;

    tileWidth=c_minPieceSize.right-c_minPieceSize.left;
    tileHeight=c_minPieceSize.bottom-c_minPieceSize.top;

    long numTileColsLong;
    tileWidth=FindTileSize(imageWidth,tileWidth,
        numTileColsNarrow,numTileColsLong);
    numTileCols=numTileColsNarrow+numTileColsLong;

    long numTileRowsLong;
    tileHeight=FindTileSize(imageHeight,tileHeight,
        numTileRowsShort,numTileRowsLong);
    numTileRows=numTileRowsShort+numTileRowsLong;

    tileRangeH=tileWidth/kSpotWidth;
    tileRangeV=tileHeight/kSpotHeight;

    numTiles=numTileCols*numTileRows;
}

void PrepareElements()
// Using tile size information, prepares the slice inventory based on average
// luminance.
// If tile size has not changed from the previous image, there is no need to redo this.
{
    if ((tileRangeH == previousTileRangeH) &&
        (tileRangeV ==
            previousTileRangeV))
        return;

    previousTileRangeH=tileRangeH;
    previousTileRangeV=tileRangeV;

    if (sliceIndex) DeleteSliceIndex();
    sliceIndex=new SlicePtr[kNumLuminanceLevels];
    std::memset(sliceIndex,0,
        kNumLuminanceLevels*sizeof(SlicePtr));
    for (int i=0;i<numElements;i++)
    {

```



```

        elementImages[i].
        PrepareSlices(sliceIndex, tileRangeH, tileRangeV);
    }

void ChooseInitialSearchRange()
// A simple attempt to define a search range that will give a good match,
// without running unreasonably long.
{
    long imageSize=desiredImage->Width()*desiredImage->Height();
    searchRange=kSearchRangeFactor/imageSize;
}

ulong MatchTile(Image& tileToMatch, SlicePtr* bestSliceP)
// Function MatchTile matches the tile against all slices within the search range.
// It starts in the slice inventory with slices of the same luminance as the tile.
// Then slices of higher and lower luminance are considered, until the search range
// is exhausted.
// For each slice, the color distance to the tile is determined,
// the smallest distance is retained as minDistance, together with bestSliceP
// Returns the minimum distance, and a pointer to the best slice in the inventory.
{
    long tileLuminance=tileToMatch.TileLuminance();
    assert(tileLuminance<kNumLuminanceLevels);
    assert(sliceIndex);

    ulong minDistance=0xFFFFFFFF;
    if (*bestSliceP) // try previous best slice first to benchmark minDistance.
    {
        minDistance=
            elementImages[(*bestSliceP->elementId)].
            RawDistance(*bestSliceP, tileToMatch, minDistance);
    }

    int leftToCheck=searchRange/2;
    for (int i=tileLuminance; i<kNumLuminanceLevels; i++)
    {
        leftToCheck=MatchSliceList(
            sliceIndex[i], tileToMatch, *bestSliceP,
            minDistance, leftToCheck);
        if (leftToCheck<=0) break;
    }

    leftToCheck=searchRange/2;
    for (int i=tileLuminance-1; i>0; i--)
    {
        leftToCheck=MatchSliceList(
            sliceIndex[i], tileToMatch, *bestSliceP,
            minDistance, leftToCheck);
        if (leftToCheck<=0) break;
    }
    return minDistance;
}

int MatchSliceList(Slice* slice, Image& tileToMatch,
                  SlicePtr& bestSlice,
                  ulong& minDistance, int leftToCheck)
// Function MatchSliceList matches slices in a list against a tile.
// Returns when the list exhausted, or earlier if the list is longer than
// the parameter "leftToCheck".
// Returns the reduced value of leftToCheck.
{
    while (slice)
    {
        ulong colorDistance=
            elementImages[slice->elementId].
            RawDistance(slice, tileToMatch, minDistance);
        if (colorDistance < minDistance)
        {
            minDistance=colorDistance;
            bestSlice=slice;
        }
        if (--leftToCheck <= 0) break;
        slice=slice->next;
    }
    return leftToCheck;
}

};

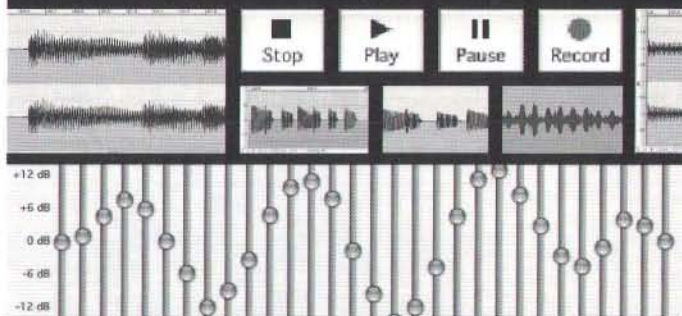
// namespace

#endif

```

audio recording • editing • effects

Felt Tip Sound Studio 2.1



FELT TIP SOFTWARE presents a MAC OS X application "SOUND STUDIO"
available on CD-ROM and as a WEB DOWNLOAD
sales by KAGI written by LUCIUS KWOK
© 2002 Felt Tip Software. All rights reserved.



www.felttip.com/ss



Impact your bottom line while working with dedicated proponents of Apple technologies

With us, your only constant is success. Our custom solutions help our clients excel their business objectives and effectively target their audience. That's why we suggest that we run your technology while you play the winning shot.

Core Competencies

- WebObjects development
- Mac OS X application development
- Cocoa application porting
- Carbon porting
- Mac OS product maintenance
- Windows/Mac OS/Unix porting
- Cross-platform development

Service Offerings

- Offshore Project Development
- On-Site Staff Augmentation
- Off-Site Project Development
- User Training

For More Information

visit <http://www.avestacs.com>
Email: mithu@avestacs.com



Avesta Computer Services, Ltd.
USA • EU • Asia



by Tim Monroe

She's Gotta Have It

Using Media Sample References and Data References

INTRODUCTION

Imagine that we've got a couple of dozen pictures from a digital camera and that we'd like to create a slide show movie from those pictures — that is, a QuickTime movie that displays each picture, in a predetermined sequence, for a predetermined amount of time. The first thing we'd need to do, of course, is create a new movie file, track, and media (by calling `CreateMovieFile`, `NewMovieTrack`, and `NewTrackMedia`). Then we might proceed like this: open each picture file, draw the picture data into an offscreen graphics world, compress the data, and add the compressed data as a new media sample by calling `AddMediaSample`. Then we would finish up by calling `InsertMediaIntoTrack` and `AddMovieResource`. Voilà, we've got our slide show movie.

This strategy involves copying the picture data from the individual picture files into the slide show movie file, and for some purposes that might be exactly what we want to do. But if we're going to keep the picture files around anyway or if we're not sure we want to keep the resulting movie, it might be better to have our QuickTime movie file just point to the data in the picture files instead of having it contain a copy of that data. We can do this by inserting into the movie a *media sample reference* (or, more briefly, a *sample reference*) to the picture file data.

In this article, we're going to work with media sample references. We'll begin by taking a look at how to create media sample references, by developing a simple droplet application that creates a slide show movie from a number of picture files, as described above. As we'll see, sample references go hand in hand with data references, which we discussed at some length in an earlier *QuickTime Toolkit* article ("Somewhere I'll Find You" in *MacTech*, October, 2000).

So we'll take this opportunity to investigate a couple of somewhat more advanced techniques for using data references. In particular, we'll see how to "flatten" a movie that contains a movie track, so that the child movie data is contained within the parent movie file; we'll also see how to create a movie whose media data is contained entirely in memory and then save it into a file.

MEDIA SAMPLE REFERENCES

In a nutshell, a media sample reference is a reference to some existing media data. The idea is that once we've got some media data stored in some location (a file, an object addressed by a URL, a block of memory, and so forth), we can reuse that media data by simply referring to it. That is, we don't need to copy the data in order to get access to it.

It's worth noting that we've bumped into sample references several times previously in this series of articles. Early on, when we discussed movie importers and exporters (in "In and Out" in *MacTech*, May, 2000), we learned that QuickTime can import some types of files without having to make a copy of the file data. We say that these kinds of files are *imported in place* — meaning that the associated movie importer constructs a movie that directly references the data in the file being imported. Now we can see that a movie importer does this by inserting media sample references into the new movie. Those references point to the data in the imported file. Importing in place, by using media sample references, allows the new movie to be created more quickly and uses less storage space (since the media data does not need to be copied).

We also ran into sample references while we were working with the QuickTime video effects architecture, when we wanted to add a video effect to part of a video track. (See "F/X 2" in *MacTech*, October 2001.) The standard way to do this is to create a video new track that is a copy of the appropriate segment of the original video track; then we create an effects track that uses the copied track segment as its input, as shown in **Figure 1**.

Tim Monroe is a member of the QuickTime engineering team. You can contact him at monroe@apple.com. The views expressed here are not necessarily shared by his employer.

Multiple formats. Multiple platforms. Complex installers.

Aladdin solves the compression and installation puzzle.

**Trying to figure out how to handle multiple
compression formats and platforms?**

The StuffIt Engine solves the compression puzzle.



Aladdin's StuffIt Engine SDK:

- Adds value to your application by integrating powerful compression and encryption.
- Is the only tool that supports the StuffIt file format.
- Provides a single API that supports over 20 compression and encoding formats common on Macintosh, Windows, and Unix.
- Makes self-extracting archives for either Macintosh or Windows.
- Available for Macintosh, Windows, Linux, or Solaris.

Licenses start as low
as \$99/year

To learn more, visit:
www.stuffit.com/sdk/

StuffIt Engine SDK™ The power of StuffIt in your software.



**Looking for the easiest and fastest
way to build an installer?**

StuffIt InstallerMaker completes your puzzle.

It's not enough just to write solid code anymore. You still have to write an installer for your users. StuffIt InstallerMaker makes it simple and effective.

- StuffIt InstallerMaker gives you all the tools you need to install, uninstall, resource-compress or update your software in one complete, easy-to-use package.
- Add marketing muscle to your installers by customizing your electronic registration form to include surveys and special offers.
- Make demoware in minutes. Create Macintosh OS X and Macintosh Classic compatible installers with StuffIt InstallerMaker.

Prices start at \$250

To learn more, visit:
[www.stuffit.com/
installermaker/](http://www.stuffit.com/installermaker/)

StuffIt InstallerMaker™ The complete installation solution.™



www.stuffit.com
(831) 761-6200

© 2002 Aladdin Systems, Inc. StuffIt, StuffIt InstallerMaker, and StuffIt Engine SDK are trademarks of Aladdin Systems, Inc. The Aladdin logo is a registered trademark. All other products are trademarks or registered trademarks of their respective holders. All Rights Reserved.

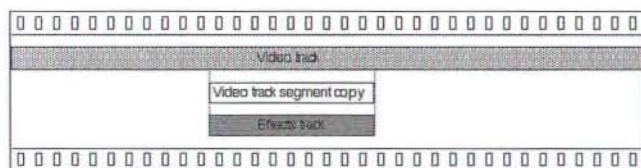


Figure 1: A filter applied to part of a video track

The new video track can contain a copy of the media data in the original video track or it can contain only references to that media data.

Creating Sample References Indirectly

Let's begin by reviewing the code we used to create the new track that we used as the input to our effects track. **Listing 1** shows the relevant section of code.

Listing 1: Creating a copy of a video track segment

```
QTEffects_AddEffectToMovieSegment
mySrcTrack1 = NewMovieTrack(theMovie, myWidth, myHeight,
    kNoVolume);
if (mySrcTrack1 == NULL)
    return(paramErr);

mySrcMedia1 = NewTrackMedia(mySrcTrack1, VideoMediaType,
    myTimeScale, NULL, 0);
if (mySrcMedia1 == NULL)
    return(paramErr);

#ifdef COPY_MOVIE_MEDIA
myErr = BeginMediaEdits(mySrcMedia1);
if (myErr != noErr)
    return(myErr);
#endif

myErr = CopyTrackSettings(myVidTrack1, mySrcTrack1);
myErr = InsertTrackSegment(myVidTrack1, mySrcTrack1,
    theStartTime, theDuration, theStartTime);
if (myErr != noErr)
    return(myErr);

#ifdef COPY_MOVIE_MEDIA
EndMediaEdits(mySrcMedia1);
#endif
```

We call `InsertTrackSegment` to copy part of the original video track (`myVidTrack1`) into the track that will be used as the source for the effect (`mySrcTrack1`). If the compiler flag `COPY_MOVIE_MEDIA` is set to 0, then we don't call `BeginMediaEdits` and `EndMediaEdits` to begin and end a media-editing session; in this case, the new track contains references to the media data in the original track, thereby minimizing the resulting file size.

Creating Sample References Directly

So, one way to create sample references is to call `InsertTrackSegment` without having opened a media-editing session (that is, without having called `BeginMediaEdits`). A more direct way to create sample references is to use the functions `AddMediaSampleReference` or `AddMediaSampleReferences`. Both of these functions allow us to add to a media one or more sample references to some existing data (and hence the names

are slightly misleading). The main difference between these two functions is that when are adding a large number of samples to a movie at one time, `AddMediaSampleReferences` is significantly more efficient than `AddMediaSampleReference`. In this article, we will be adding only one sample reference at a time, so we'll restrict our attention to `AddMediaSampleReference`, which is declared essentially like this:

```
OSErr AddMediaSampleReference (
    Media theMedia,
    long dataOffset,
    unsigned long size,
    TimeValue durationPerSample,
    SampleDescriptionHandle sampleDescriptionH,
    long numberOfSamples,
    short sampleFlags,
    TimeValue *sampleTime);
```

The parameters here are identical to those for `AddMediaSample`, with one exception: `AddMediaSample` also takes a handle to the data that is to be added to the media. With `AddMediaSampleReference`, we're not adding any data, so we don't need that parameter.

Here's how we might call `AddMediaSampleReference` to add a single media sample reference to a media:

```
myErr = AddMediaSampleReference(myMedia, myDataOffset,
    mySize, myDuration, myDesc, 1, 0, NULL);
```

As you can see, the `numberOfSamples` parameter is set to 1, the `sampleFlags` parameter is set to 0, and the `sampleTime` parameter is set to `NULL` (since we don't care to have the new sample time returned to us). The other parameters are set to specific values determined by the application. For example, `myDataOffset` should specify the offset into the referenced media file (or other storage device) of the desired media data. In our sample slide show making application, `myDuration` will always be set to 600, so that each slide is displayed for one second.

Now, how does `AddMediaSampleReference` know where the original media data resides? As you know, QuickTime uses data references as its principal means of identifying the location of some data. So we might have expected that `AddMediaSampleReference` would take a data reference as a parameter. But, alas, you can see above that there is no such parameter. Instead, we need to attach the data reference to the media before we call `AddMediaSampleReference`; we do this by calling the `AddMediaDataRef` function, like this:

```
myErr = AddMediaDataRef(myMedia, &myDataRefIndex,
    (Handle)myAlias, rAliasType);
```

`AddMediaDataRef` adds the specified data reference to the specified media and returns the index of that data reference in the media's list of data references. Then we assign that index to the `dataRefIndex` field of the sample description that we pass to `AddMediaSampleReference`:

```
(**myDesc).dataRefIndex = myDataRefIndex;
```


In this way, we've given `AddMediaSampleReference` a complete specification of the location of the data for which it will create a sample reference.

Getting Sample References

QuickTime also provides the functions `GetMediaSampleReference` and `GetMediaSampleReferences`, which we can use to get information about one or more samples that are stored in a media data file (or other media storage container). Unlike `GetMediaSample`, `GetMediaSampleReference` does not return the actual media data to us; rather, it gives us a handful of pieces of information about the media sample(s), such as the offset within the media container of the sample data, the size of the sample data, and a sample description that specifies the format of that sample data. We saw just above that we'll need that offset and size when we call `AddMediaSampleReference`; we'll also need a sample description when we're building our slide show movie. So we'll call `GetMediaSampleReference` like this:

```
myErr = GetMediaSampleReference(myRefMedia,
    &myDataOffset, &mySize, 0, NULL, NULL, myDesc,
    NULL, 1, NULL, 0);
```

Here, `myRefMedia` is the media to which we have a reference. In our slide show example, it'll be the data in the individual picture files.

SLIDE SHOW MOVIES

Let's illustrate how to work with media sample references by building a slide show movie from a collection of picture files. We want the user to be able to drop any number of picture files onto our application; when this happens, the application will ask the user to specify a file name and location for the slide show movie; then it will create the movie and exit. Let's call this droplet `DropPix`.

Handling Dropped Files

The first thing that `DropPix` needs to do is assemble a list of the files that the user has dropped onto its icon. In our Macintosh application, we can do this inside of our AppleEvent handler for the Open Document event, as shown in **Listing 2**.

Listing 2: Keeping track of dropped files (Macintosh)

```
QTApp_HandleOpenDocumentAppleEvent
// open each specified file
for (myIndex = 1; myIndex <= myItemsInList; myIndex++)
    if (myErr == noErr) {
        myErr = AEGGetNthPtr(&myDocList, myIndex, typeFSS,
            &myKeyWd, &myTypeCode, (Ptr)&myFSSpec,
            sizeof(myFSSpec), &myActualSize);
        if (myErr == noErr) {
            gSpecs[myIndex - 1] = myFSSpec;
        }
    }

if (myDocList.dataHandle)
    myIgnoreErr = AEDisposeDesc(&myDocList);

gNumSpecs = myIndex - 1;
DropPix_MakeSlideShow();

QTFrame_QuitFramework(); // act like a droplet and close automatically
```

You'll notice that we're using two global variables, `gSpecs` and `gNumSpecs`, to keep track the file system specification records for the dropped files. We declare those variables like this:

```
FSSpec    gSpecs[kMaxNumPictureFiles];
short     gNumSpecs;
```

(I'll leave it as an exercise for the enterprising reader to get rid of the hard-coded array size.)

On Windows, we can get a list of the dropped files by reworking some of the code in the `QTFrame_OpenCommandLineMovies` function (defined in the file `WinFramework.c`). First we need to remove the existing call to `SHGetFileInfo` that restricts our application to opening only QuickTime movie files. Then, once we've finished creating a

Valentina

Object-Relational SQL Database

The fastest database engine for MacOS/Windows

It operates 100's and sometimes
a 1000 times faster than other systems

www.paradigmasoft.com

Hosted by macserve.net

Download full featured evaluation version



Get more out of your Machine

**The place to buy tools & toys
on the cutting edge of Macintosh.**

BLUETOOTH ADAPTER for MAC OS X

Use your Cell Phone as a Modem

Imagine jumping in a cab, opening your PowerBook and browsing the Internet. Your Internet connection is the cell phone, still in your pocket and with no wires attached. Imagine synching your PDA to your office computer each day, even if you forget to take it out of your coat pocket.

Bluetooth is an open specification for short-range wireless connections between desktop and laptop computers, personal digital assistants, cellular phones, printers, scanners, digital cameras and home appliances. With Mac OS X, Bluetooth is now part of the Mac operating system.

The Bluetooth adapter is a thumb sized device that plugs into your USB port and let's Mac OS X talk to other Bluetooth devices – like the HP 995c printer or Sony Ericsson T68i cell phone. The Bluetooth USB Adapter is in stock, ready to ship, for only \$49.95.

To order or learn more, visit www.devdepot.com/bluetooth

THE NEW REALbasic

Anyone can create applications for Mac & Windows!

Building your applications is easier than you imagine. REALbasic is simply the easiest way to create your own stand alone applications for Mac OS 9, Mac OS X, even Windows. Professional developers will be amazed at the power and flexibility of the new 4.5 release. REALbasic Pro's sophisticated version of Object-Oriented BASIC let's you quickly create database front ends, prototype new concepts in minutes, or build entire commercial applications.

REALbasic is a power tool that's easy to use. Each window type, control, and menu is preconfigured and instantly works as it should. The drag-and-drop Window Editor allows you to quickly and easily create your application's interface so you can focus on the important part – your creativity.

To order or learn more, visit www.devdepot.com/realbasic

www.devdepot.com





PUT A FIREWIRE CAMERA on YOUR POWERBOOK

Includes free video conferencing software!

It's called Fire-i – a compact, crystal clear, video camera that clips onto your PowerBook display or sits on your desktop. FireWire based, its blazing fast 400 Mbps rate (more than 30 times USB) and is ideal for video conferencing. Perfect for PowerBooks, the smart attachment clip, compact size and low power consumption makes it easy to use anywhere.

Fire-i is natively supported by Jaguar (the latest Mac OS X) and runs on Mac OS 9 or above. It includes free, easy to use software for video conferencing – the ideal way to stay in touch with students heading off to school, co-workers in the field, or family and friends.

To order or learn more, visit www.devdepot.com/fire-i

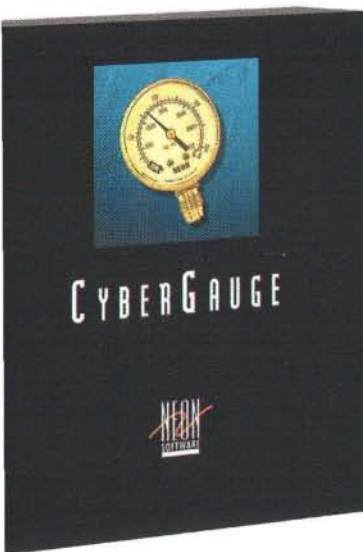
USE AIRPORT with ANY USB MACINTOSH

MacWireless USB Adapter for 802.11b

Apple's AirPort technology (802.11b) lets you connect computers to each other, to printers, or to the internet easily and without wires. It's easy to install an 802.11b router to share internet access or printing and Apple's AirPort pod is a great way to use dial-up services – but what do you do if your Mac doesn't have AirPort support?

DevDepot is featuring the first USB AirPort (802.11b) adapter for Macintosh. For only \$119 you can add AirPort support to your iMac or other USB Macintosh without opening the case or installing any hardware – just plug it in and it works.

To order or learn more, visit www.devdepot.com/airport



KNOW YOUR NETWORK

CyberGauge shows you in real-time how your network works!

Are you getting the bandwidth you're paying for? Micro outages and denial of service attacks can cost your business money, in some cases without you ever knowing the cause.

CyberGauge is an essential network monitoring tool that monitors bandwidth usage by machines in your network. It can sense a denial-of-service attack before it gets critical or micro outages that you never knew were disrupting your eCommerce systems – and eMail you when it happens. If you're paying big money for internet connectivity, you should consider CyberGauge – it will quickly pay for itself.

To order or learn more, visit www.devdepot.com/cybergauge

file system specification record for a dropped file, we can add it to our array and increment our count of dropped files, as shown in **Listing 3**.

Listing 3: Keeping track of dropped files (Windows)

```

// make an FSSpec record
NativePathNameToFSSpec(myFileName, &myFSSpec,
    kFullNativePath);
gSpecs[myFileIndex] = myFSSpec;
myFileIndex++;

When we are done collecting the files, we finish up like this:
gNumSpecs = myFileIndex;
DropPix_MakeSlideShow();

```

Creating the Slide Show Movie

The `DropPix_MakeSlideShow` function first elicits a movie file name and location from the user, so that `DropPix` knows where to put the output slide show movie. Then it creates the new movie file:

```

myErr = CreateMovieFile(&myFile, sigMoviePlayer,
    smCurrentScript, myFlags, &myResRefNum, &myMovie);

```

And, as usual, we'll create a new track and media:

```

myTrack = NewMovieTrack(myMovie, myWidth, myHeight, 0);
myMedia = NewTrackMedia(myTrack, VideoMediaType, 600, NULL,
    0);

```

The bulk of `DropPix_MakeSlideShow` is a for loop that adds to this new media a sample reference to the data in each of the picture files in `gSpecs`.

Retrieving the Picture Information

Before we can call `AddMediaSampleReference`, we need to get the offset of the picture data in its file, and we need to add a data reference for that file to the new media. Adding a data reference is easy; first we create a file data reference:

```

myErr = QTNewAlias((const FSSpec *)&gSpecs[myIndex],
    &myAlias, true);

```

Then we add it to the media:

```

myErr = AddMediaDataRef(myMedia, &myDataRefIndex,
    (Handle)myAlias, rAliasType);

```

How do we get the offset of the data in the picture file? Earlier, we saw that we could use `GetMediaSampleReference` to get information about samples that are stored in a media data file. So all we need to do is call `NewMovieFromDataRef` to open the picture file as a movie and then call `GetMediaSampleReference` on that movie's media. **Listing 4** shows the sequence of calls here.

Listing 4: Getting information about a picture file

```

DropPix_MakeSlideShow

// allocate a sample description
myDesc = (SampleDescriptionHandle)NewHandle(0);
myErr = MemError();
if (myErr != noErr)
    goto bailLoop;

myErr = NewMovieFromDataRef(&myRefMovie,
    newMovieDontResolveDataRefs, NULL, (Handle)myAlias,

```

```

    rAliasType);
if (myErr != noErr)
    goto bailLoop;

// get the first track's media
myRefTrack = GetMovieIndTrack(myRefMovie, 1);
myRefMedia = GetTrackMedia(myRefTrack);
if ((myRefTrack == NULL) || (myRefMedia == NULL))
    goto bailLoop;

myErr = GetMediaSampleReference(myRefMedia, &myDataOffset,
    &mySize, 0, NULL, NULL, myDesc, NULL, 1, NULL, 0);

```

This gives us the data offset and the media sample size. It also gives us a sample description of the image, which we can use (for instance) to determine the size of the video track in the slide show movie. Our call to `NewMovieTrack` really looks like this:

```

myTrack = NewMovieTrack(myMovie,
    Long2Fix((*(ImageDescriptionHandle)myDesc).width),
    Long2Fix((*(ImageDescriptionHandle)myDesc).height),
    kNoVolume);

```

Adding a Sample Reference

Now we're almost finished. We add a sample reference to the data in the picture file like this:

```

(**myDesc).dataRefIndex = myDataRefIndex;
myErr = AddMediaSampleReference(myMedia, myDataOffset,
    mySize, myDuration, myDesc, 1, 0, NULL);

```

Once we've done this for each picture file dropped onto our application, we need to call `InsertMediaIntoTrack` and `AddMovieResource` in the usual way. Listing 5 shows `DropPix_MakeSlideShow` in its full glory.

Listing 5: Creating a slide show movie

```

DropPix_MakeSlideShow

OSErr DropPix_MakeSlideShow (void)
{
    Movie      myMovie = NULL;
    Track      myTrack = NULL;
    Media      myMedia = NULL;
    FSSpec      myFile;
    Boolean     myIsSelected = false;
    Boolean     myIsReplacing = false;
    StringPtr   myPrompt =
        QTUtils_ConvertCToPascalString("Save movie as:");
    StringPtr   myFileName =
        QTUtils_ConvertCToPascalString("Untitled.mov");
    long        myFlags = createMovieFileDeleteCurFile |
        createMovieFileDontCreateResFile;
    short       myResRefNum = kInvalidFileRefNum;
    short       myResID = movieInDataForkResID;
    short       myIndex;
    OSErr       myErr = noErr;

    if (gNumSpecs <= 0)
        return(paramErr);

    // prompt the user for new file name
    QTFrame_PutFile(myPrompt, myFileName, &myFile,
        &myIsSelected, &myIsReplacing);
    myErr = myIsSelected ? noErr : userCanceledErr;
    if (myErr != noErr)
        goto bail;

    // delete any existing file of that name
    if (myIsReplacing) {
        myErr = DeleteMovieFile(&myFile);
    }
}

```



```

if (myErr != noErr)
    goto bail;
}

// create a movie file for the destination movie
myErr = CreateMovieFile(&myFile, sigMoviePlayer,
    smCurrentScript, myFlags, &myResRefNum, &myMovie);
if (myErr != noErr)
    goto bail;

// add a sample reference for each image to the new movie
for (myIndex = 0; myIndex < gNumSpecs; myIndex++) {
    short    myDataRefIndex = 0;
    long     myDataOffset, mySize;
    SampleDescriptionHandle
        myDesc = NULL;
    TimeValue myDuration = 600;
    AliasHandle myAlias = NULL;
    Movie      myRefMovie = NULL;
    Track      myRefTrack = NULL;
    Media      myRefMedia = NULL;

    myErr = QTNewAlias((const FSSpec *)&gSpecs[myIndex],
        &myAlias, true);
    if (myErr != noErr)
        goto bailLoop;

    // allocate sample description
    myDesc = (SampleDescriptionHandle)NewHandle(0);
    myErr = MemError();
    if (myErr != noErr)
        goto bailLoop;

    myErr = NewMovieFromDataRef(&myRefMovie,
        newMovieDontResolveDataRefs, NULL, (Handle)myAlias,
        rAliasType);
    if (myErr != noErr)
        goto bailLoop;

    // get the first track's media
    myRefTrack = GetMovieIndTrack(myRefMovie, 1);

```

```

myRefMedia = GetTrackMedia(myRefTrack);
if ((myRefTrack == NULL) || (myRefMedia == NULL))
    goto bailLoop;

myErr = GetMediaSampleReference(myRefMedia,
    &myDataOffset, &mySize, 0, NULL, NULL, myDesc,
    NULL, 1, NULL, 0);
if (myErr != noErr)
    goto bailLoop;

if (myTrack == NULL) {
    // create the movie track and media
    myTrack = NewMovieTrack(myMovie,
        Long2Fix((** (ImageDescriptionHandle)myDesc).width),
        Long2Fix((** (ImageDescriptionHandle)myDesc).height),
        kNoVolume);
    myErr = GetMoviesError();
    if (myErr != noErr)
        goto bail;

    myMedia = NewTrackMedia(myTrack, VideoMediaType, 600,
        NULL, 0);
    myErr = GetMoviesError();
    if (myErr != noErr)
        goto bail;

    // add a data reference to the media
    myErr = AddMediaDataRef(myMedia, &myDataRefIndex,
        (Handle)myAlias, rAliasType);
    if (myErr != noErr)
        goto bailLoop;

    (**myDesc).dataRefIndex = myDataRefIndex;

    // add a media sample reference to the media
    myErr = AddMediaSampleReference(myMedia, myDataOffset,
        mySize, myDuration, myDesc, 1, 0, NULL);

    bailLoop;
    if (myDesc)

```

United States Postal Service

Statement of Ownership, Management, and Circulation

| | | | |
|---|--|--|--|
| 1. Publication Title MacTech Magazine | | 2. Publication Number 1067-8360 | 3. Filing Date 10/1/02 |
| 4. Issue Frequency Monthly | | 5. Number of Issues Published Annually 12 | 6. Annual Subscription Price \$47.00 |
| 7. Complete Mailing Address of Known Office of Publication (Not printer) (Street, city, county, state, and ZIP+4) Post Office Box 5200 Westlake Village CA 91359-5200 | | | |
| 8. Complete Mailing Address of Headquarters or General Business Office of Publisher (Not printer) Same as above | | 9. Contact Person Sam Webber Telephone 805 491 9797 | |
| 10. Full Names and Complete Mailing Addresses of Publisher, Editor, and Managing Editor (Do not leave blank) | | | |
| Publisher (Name and complete mailing address) Deil Ticklin PO Box 5200 Westlake Village CA 91359-5200 | | | |
| Editor (Name and complete mailing address) Dave Mark (after October 03) PO Box 5200 Westlake Village CA 91359-5200 | | | |
| Managing Editor (Name and complete mailing address) Jessica Stubbelfield PO Box 5200 Westlake Village CA 91359-5200 | | | |
| 11. Owner (Do not leave blank. If the publication is owned by a corporation, give the name and address of the corporation immediately followed by the names and addresses of all stockholders owning or holding 1 percent or more of the total amount of stock. If not owned by a corporation, give the names and addresses of the individual owners. If owned by a partnership or other unincorporated firm, give its name and address as well as those of each individual owner. If the publication is published by a nonprofit organization, give its name and address.) | | | |
| 12. Name Plain Corporation | | Complete Mailing Address PO Box 5200 Westlake Village CA 91359 | |
| 13. Name Andrea Sniderman | | Address Above | |
| 14. Name Deil Ticklin | | Address Above | |
| 15. Known Bondholders, Mortgagees, and Other Security Holders Owning or Holding 1 Percent or More of Total Amount of Bonds, Mortgages, or Other Securities. If none, check box <input checked="" type="checkbox"/> None | | | |
| 16. Name | | Complete Mailing Address | |
| | | | |
| | | | |
| | | | |
| | | | |
| 17. Tax Status (For completion by nonprofit organizations authorized to mail at nonprofit rates) (Check one) <input checked="" type="checkbox"/> Has Not Changed During Preceding 12 Months <input type="checkbox"/> Has Changed During Preceding 12 Months (Publisher must submit explanation of change with this statement) | | | |

Form 3526, October 1999

(See Instructions on Reverse)

| | | | |
|---|--|--|---|
| 13. Publication Title MacTech Magazine | | 14. Issue Date for Circulation Data Below August, 2002 | |
| 15. Extent and Nature of Circulation | | Average No. Copies Each Issue During Preceding 12 Months | No. Copies of Single Issue Published Nearest to Filing Date |
| a. Total Number of Copies (Net press run) | | 8606 | 6497 |
| b. Paid and/or Requested Circulation | | | |
| (1) Paid (Requested Outside-County Mail Subscriptions Stated on Form 3541, (Include advertiser's proof and exchange copies)) | | 4854 | 3667 |
| (2) Paid In-County Subscriptions Stated on Form 3541 (Include advertiser's proof and exchange copies) | | 0 | 0 |
| (3) Sales Through Dealers and Carriers, Street Vendors, Counter Sales, and Other Non-USPS Paid Distribution | | 2237 | 1498 |
| (4) Other Classes Mailed Through the USPS | | 2 | 0 |
| c. Total Paid and/or Requested Circulation (Sum of 15b (1), (2), (3), and (4)) | | 7093 | 5165 |
| d. Free Distribution by Mail (Samples, complimentary, and other free) | | | |
| (1) Outside-County as Stated on Form 3541 | | 0 | 0 |
| (2) In-County as Stated on Form 3541 | | 0 | 0 |
| (3) Other Classes Mailed Through the USPS | | 37 | 11 |
| e. Free Distribution Outside the Mail (Carriers or other means) | | 426 | 222 |
| f. Total Free Distribution (Sum of 15d and 15e.) | | 463 | 233 |
| g. Total Distribution (Sum of 15c and 15f) | | 7556 | 5398 |
| h. Copies Not Distributed | | 1050 | 99 |
| i. Total (Sum of 15g and h.) | | 8606 | 6497 |
| j. Percent Paid and/or Requested Circulation (15c divided by 15g times 100) | | 94 | 96 |
| 16. Publication of Statement of Ownership <input checked="" type="checkbox"/> Publication required. Will be printed in the November issue of this publication. <input type="checkbox"/> Publication not required. | | | |
| 17. Signature and Title of Editor, Publisher, Business Manager, or Owner managing editor | | Date 10/2/02 | |

I certify that all information furnished on this form is true and complete. I understand that anyone who furnishes false or misleading information on this form or who omits material or information requested on the form may be subject to criminal sanctions (including fines and imprisonment) and/or civil sanctions (including civil penalties).

Instructions to Publishers

- Complete and file one copy of this form with your postmaster annually on or before October 1. Keep a copy of the completed form for your records.
- In cases where the stockholder or security holder is a trustee, include in items 10 and 11 the name of the person or corporation for whom the trustee is acting. Also include the names and addresses of individuals who are stockholders who own or hold 1 percent or more of the total amount of bonds, mortgages, or other securities of the publishing corporation. In item 11, if none, check the box. Use blank sheets if more space is required.
- Be sure to furnish all circulation information called for in item 15. Free circulation must be shown in items 15d, e, and f.
- Item 15h., Copies Not Distributed, must include (1) newspaper copies originally stated on Form 3541, and returned to the publisher, (2) estimated returns from news agents, and (3) copies for office use, leftovers, spoiled, and all other copies not distributed.
- If the publication had Periodicals authorization as a general or requester publication, this Statement of Ownership, Management, and Circulation must be published; it must be printed in any issue in October or, if the publication is not published during October, the first issue printed after October.
- In item 16, indicate the date of the issue in which this Statement of Ownership will be published.
- Item 17 must be signed.

Failure to file or publish a statement of ownership may lead to suspension of Periodicals authorization.


```

    DisposeHandle((Handle)myDesc);
}

// add the media to the track
myErr = InsertMediaIntoTrack(myTrack, 0, 0,
    GetMediaDuration(myMedia), fixed1);
if (myErr != noErr)
    goto bail;

// add the movie atom to the movie file
myErr = AddMovieResource(myMovie, myResRefNum, &myResID,
    NULL);

bail:
    if (myResRefNum != kInvalidFileRefNum)
        CloseMovieFile(myResRefNum);

    if (myMovie != NULL)
        DisposeMovie(myMovie);

    free(myPrompt);
    free(myFileName);

    return(myErr);
}

```

As you can see, the size of the slide show video track is determined by the size of the first picture file in `gSpecs`. Each remaining image is scaled to fit into that track rectangle, which may result in some distortion of the image. I'll leave it as an exercise for the reader to figure out a way to avoid that distortion.

MOVIE TRACKS

A movie track is a track of type `MovieMediaType` that effectively embeds one movie inside of another, as illustrated in **Figure 2**. The key feature of using movie tracks — instead of just layering one track on top of another track — is that the parent and child movies can have different time bases, so they can (for instance) have different playback rates and different looping characteristics.

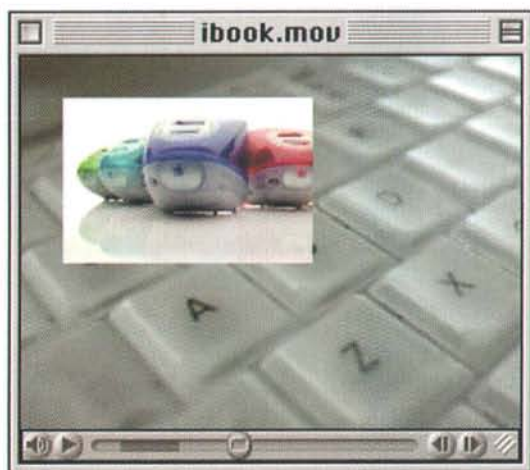


Figure 2: A child movie inside of a parent movie

You may recall from our earlier discussion of movie tracks (in “Atomic Café” in *MacTech*, September 2000) that a media sample in a movie track consists of an atom container whose

atoms specify the movie that is to be embedded in the main movie, as well as some of the playback characteristics of the embedded movie. The media sample does not typically contain the data for the child movie itself; rather, it points to that data using an atom of type `kMovieMediaDataReference`; this atom contains a data reference to the child movie data, which is most often a file data reference or a URL data reference.

It would be nice if there were a way to save a movie that contains a movie track, so that all the media data — for the parent *and* the child movies — is contained within a single file. This is actually quite straightforward, using a data reference extension. The essential idea is to simply append all of the child movie data to the data reference atom in the appropriate media sample. By suitably reconfiguring the data reference atom, we can force QuickTime to look in the data reference extension for the child movie data instead of resolving the data reference in that atom.

In the process of reconfiguring the data reference atom, we'll need to use a few techniques that are interesting in their own right. We'll need to learn how to get the data for a media sample at a specific movie time, and we'll need to learn how to replace an entire media sample in an existing movie. So let's get started.

Getting the Current Media Sample

A movie track (that is, a track of type `MovieMediaType`) can have one or more media samples, each of which is an atom container whose atoms pick out a child movie and specify its spatial layout and playback characteristics. We can get the media sample for a given movie time by calling `GetMediaSample`. The only “gotcha” is that `GetMediaSample` requires that this time be expressed in the media's time scale. To convert a movie time to the corresponding media time, we can call `TrackTimeToMediaTime`, as illustrated in **Listing 6**.

Listing 6: Getting the current media sample data

```

QTMM FlattenChildIntoParent
Handle mySample = NULL;
SampleDescriptionHandle myDesc = NULL;
TimeValue myMovieTimeNow = 0;
TimeValue myMediaTimeNow = 0;
TimeValue myDuration = 0;

mySample = NewHandleClear(0);
if (mySample == NULL)
    return(MemError());

myDesc = (SampleDescriptionHandle)NewHandleClear(0);
if (myDesc == NULL) {
    myErr = MemError();
    goto bail;
}

myMovieTimeNow = GetMovieTime(myMovie, NULL);
if (myMovieTimeNow == GetMovieDuration(myMovie))
    myMovieTimeNow--;

myMediaTimeNow = TrackTimeToMediaTime(myMovieTimeNow,
    myTrack);
if (myMediaTimeNow == -1) {
    myErr = invalidTime;
    goto bail;
}

```


Increase your projectivity.



Be productive. Stay productive.

Introducing FastTrack Schedule 8. Redesigned for Mac OS X and packed with new productivity features and a bold Aqua interface—FastTrack Schedule 8 has all the tools you need to ensure project success. For a free demo version or to order, call us today at 800.450.1983.

www.fasttrackschedule8.com




```
myErr = GetMediaSample(myMedia, mySample, 0, NULL,
    myMediaTimeNow, NULL, &myDuration, myDesc, NULL, 1,
    NULL, NULL);
```

TrackTimeToMediaTime returns `-1` if there is no media sample in the track at the specified movie time or if the specified movie time is outside the movie's active segment. Since we call **GetMovieTime** to get the current movie time, we're guaranteed that **myMovieTimeNow** will be within the active movie segment. (Notice that we decrement the movie time if we happen to be at the end of the movie.)

The time passed to **GetMediaSample** can be any time within the extent of the media sample. **GetMediaSample** retrieves the data for that sample and returns it in the handle we pass it (here, **mySample**). **GetMediaSample** also returns the duration of the media sample.

Loading the Child Movie Data into Memory

The media data, to repeat, is an atom container that holds at least a data reference atom that picks out the child movie. We can get that atom like this:

```
myAtom = QTFindChildByIndex(mySample,
    kParentAtomIsContainer, kMovieMediaDataReference,
    1, NULL);
```

If **myAtom** is non-zero, then we want to fetch the data in that atom. The data is a data reference prefixed by the data reference type. We can get the data reference and its type like this:

```
QTGetAtomDataPtr(mySample, myAtom, &myDataSize,
    &myDataPtr);
myDataRefType = EndianU32_BtoN(*(OSType *)myDataPtr);
myErr = PtrToHand(myDataPtr + sizeof(OSType),
    &myDataRef, myDataSize - sizeof(OSType));
```

If this is all successful, then **myDataRefType** is the data reference type and **myDataRef** is the data reference itself. Right now, we want to open the child movie specified by that data reference and load it completely into memory. To open the child movie, we can use **NewMovieFromDataRef**:

```
NewMovieFromDataRef(&myChildMovie, 0, NULL, myDataRef,
    myDataRefType);
```

To load the movie's media data completely into memory, we can use **FlattenMovieData**, passing it a handle data reference. (We've used this trick previously, in "Somewhere I'll Find You", cited earlier.) **Listing 7** shows the essential steps.

Listing 7: Loading a child movie into memory

```
QTMMIM_FlattenChildIntoParent
DataReferenceRecord    myDataRefRecord;
Handle                 myDataRefHandle = NULL;
Handle                 myHandleDataRef = NULL;

myDataRefHandle = NewHandleClear(0);
if (myDataRefHandle == NULL)
    goto bail;

myHandleDataRef = QTDR_MakeHandleDataRef(myDataRefHandle);
if (myHandleDataRef == NULL)
    goto bail;
```

```
myDataRefRecord.dataRefType = HandleDataHandlerSubType;
myDataRefRecord.dataRef = myHandleDataRef;
```

```
myMemoryMovie = FlattenMovieData(myChildMovie,
    flattenFSSpecPtrIsDataRefRecordPtr |
    flattenAddMovieToDataFork,
    (FSSpecPtr)&myDataRefRecord,
    sigMoviePlayer,
    smSystemScript,
    0L);
```

```
myErr = GetMoviesError();
if (myErr != noErr)
    goto bail;
```

```
DisposeMovie(myChildMovie);
```

Notice that the second parameter passed to **FlattenMovieData** contains the **flattenFSSpecPtrIsDataRefRecordPtr** flag, which indicates that the third parameter is a pointer to a data reference record, not a pointer to a file system specification record; it also contains the **flattenAddMovieToDataFork** flag, which tells **FlattenMovieData** to write the movie atom as well as the media data into the specified location. If we didn't specify **flattenAddMovieToDataFork**, we'd get only the media data in the child movie.

Creating a "Flattened" Child Movie Media Sample

Recall that we want to replace the original data reference in the data reference atom in the child movie media sample by a new data reference that has the child movie data appended to it. That is to say, we want to attach a data reference extension to that original data reference. In this case, the extension is of type **kDataRefExtensionInitializationData**. Let's call this kind of extension an *initialization data data reference extension* — or, more briefly, an *initialization extension*.

QuickTime uses an initialization extension in one case only: when the data reference is a handle data reference and the specified handle is **NULL**. When this happens, QuickTime takes the data directly from the initialization extension. In effect, we can use this type of data reference extension to short-circuit the normal data reference resolution that QuickTime would otherwise perform; we're saying: here is the data you're looking for, in this data reference extension.

So our task boils down to this: replace the existing data reference in the atom of type **kMovieMediaDataReference** by a handle data reference whose associated data is **NULL**; then append a data reference extension of type **kDataRefExtensionInitializationData** to that data reference. For this latter task, we'll use the utility function **QTDR_AddInitDataDataRefExtension**, defined in **Listing 8**.

Listing 8: Adding an initialization data data reference extension

```
QTDR_AddInitDataDataRefExtension
OSErr QTDR_AddInitDataDataRefExtension
    (Handle theDataRef, Ptr theInitDataPtr)
{
    unsigned long myAtomHeader[2];
    OSErr myErr = noErr;

    if (theInitDataPtr == NULL)
        return(paramErr);

    myAtomHeader[0] = EndianU32_NtoB(sizeof(myAtomHeader) +
        GetPtrSize(theInitDataPtr));
```


**"Without a doubt, the Premiere Resource Editor
for the Mac OS ... A wealth of time-saving tools."**

– MacUser Magazine Eddy Awards

"A distinct improvement over Apple's ResEdit."

– MacTech Magazine

"Every Mac OS developer should own a copy of Resorcerer."

– Leonard Rosenthal, Aladdin Systems

**"Without Resorcerer, our localization efforts would look like a
Tower of Babel. Don't do product without it!"**

– Greg Galanos, CEO and President, Metrowerks

"Resorcerer's data template system is amazing."

– Bill Goodman, author of *Smaller Installer* and *Compact Pro*

"Resorcerer Rocks! Buy it, you will NOT regret it."

– Joe Zobkiw, author of *A Fragment of Your Imagination*

"Resorcerer will pay for itself many times over in saved time and effort."

– MacUser review

"The template that disassembles 'PICT's is awesome!"

– Bill Steinberg, author of *Pyro!* and *PBTools*

"Resorcerer proved indispensable in its own creation!"

– Doug McKenna, author of *Resorcerer*



Resorcerer® 2

Version 2.0

The Resource Editor for the Mac™ OS Wizard

ORDERING INFO

Requires System 7.0 or greater,
1.5MB RAM, CD-ROM

Standard price: \$256 (decimal)

Website price: \$128 - \$256

(Educational, quantity, or
other discounts available)

Includes: Electronic documentation
60-day Money-Back Guarantee
Domestic standard shipping

Payment: Check, PO's, or Visa/MC
Taxes: Colorado customers only

Extras (call, fax, or email us):
COD, FedEx, UPS Blue/Red,
International Shipping

MATHEMAESTHETICS, INC.
PO Box 298
Boulder, CO 80306-0298 USA
Phone: (303) 440-0707
Fax: (303) 440-0504
resorcerer@mathemaesthetics.com

**New
in
2.0:**

- Very fast, HFS browser for viewing file tree of all volumes
- Extensibility for new Resorcerer Apprentices (CFM plug-ins)
- New AppleScript Dictionary ('aete') Apprentice Editor
- MacOS 8 Appearance Manager-savvy Control Editor
- PowerPlant text traits and menu command support
- Complete AIFF sound file disassembly template
- Big-, little-, and even mixed-endian template parsing
- Auto-backup during file saves; folder attribute editing
- Ships with PowerPC native, fat, and 68K versions

- Fully supported; it's easier, faster, and more productive than ResEdit
- Safer memory-based, not disk-file-based, design and operation
- All file information and common commands in one easy-to-use window
- Compares resource files, and even **edits your data forks** as well
- Visible, accumulating, editable scrap
- Searches and opens/marks/selects resources by text content
- Makes global resource ID or type changes easily and safely
- Builds resource files from simple Rez-like scripts
- Most editors DeRez directly to the clipboard
- All graphic editors support screen-copying or partial screen-copying
- Hot-linking Value Converter for editing 32 bits in a dozen formats
- Its own 32-bit List Mgr can open and edit very large data structures
- Templates can pre- and post-process any arbitrary data structure
- Includes nearly 200 templates for common system resources
- TMPLs for Installer, MacApp, QT, Balloons, AppleEvent, GX, etc.
- Full integrated support for editing color dialogs and menus
- Try out balloons, 'ictb's, lists and popups, even create C source code
- Integrated single-window Hex/Code Editor, with patching, searching
- Editors for cursors, versions, pictures, bundles, and lots more
- Relied on by thousands of Macintosh developers around the world

To order by credit card, or to get the latest news, bug fixes, updates, and apprentices, visit our website...

www.mathemaesthetics.com


```

myAtomHeader[1] = EndianU32_NtoB(
    kDataRefExtensionInitializationData);

myErr = PtrAndHand(myAtomHeader, theDataRef,
    sizeof(myAtomHeader));
if (myErr == noErr)
    myErr = PtrAndHand(theInitDataPtr, theDataRef,
        GetPtrSize(theInitDataPtr));

return(myErr);
}

```

Listing 9 shows the code we use to replace the original data reference atom by a new data reference atom that contains the data of the child movie.

Listing 9: Replacing a data reference atom by a “flattened” atom

QTMIM_FlattenChildIntoParent

```

// out with the old...
QTRemoveAtom(mySample, myAtom);

// ...and in with the new
myChildDataHandle = NewHandleClear(sizeof(OSType) +
    sizeof(Handle));
if (myChildDataHandle != NULL) {
    OSType myType;

    // set the data reference type
    myType = EndianU32_NtoB(HandleDataHandlerSubType);
    BlockMove(&myType, *myChildDataHandle, sizeof(OSType));

    // leave the next four bytes set to 0x00000000;
    // add a filenaming extension and an initialization extension
    myErr = QTDR_AddFilenamingExtension(myChildDataHandle,
        NULL);
    if (myErr != noErr)
        goto bail;

    HLock(myDataRefHandle);
    myErr = QTDR_AddInitDataDataRefExtension(
        (myChildDataHandle, *myDataRefHandle);
    HUnlock(myDataRefHandle);
    if (myErr != noErr)
        goto bail;

    HLock(myChildDataHandle);
    myErr = QTInsertChild(mySample, kParentAtomIsContainer,
        kMovieMediaDataReference, 1, 1,

```

```

    GetHandleSize(myChildDataHandle),
    *myChildDataHandle, NULL);
    HUnlock(myChildDataHandle);
    if (myErr != noErr)
        goto bail;
}

```

Notice that we need to add an empty filenaming extension before we add the initialization extension.

Replacing a Media Sample

One final step remains, namely to replace the original media sample by the revised media sample. To do this, we first need to delete the track segment corresponding to the original media sample. Then we'll add the new media sample into the media and insert it into the track at that time.

It's easy enough to figure out the start time and duration of a particular media sample. Recall that we already have the current movie time stored in the variable `myMovieTimeNow`. We can then call `GetTrackNextInterestingTime` twice, asking it to search backward and forward for the boundaries of the current media sample:

```

GetTrackNextInterestingTime(myTrack,
    nextTimeMediaSample | nextTimeEdgeOK,
    myMovieTimeNow, -0x01000, &myMovieStartTime, NULL);
GetTrackNextInterestingTime(myTrack,
    nextTimeMediaSample | nextTimeEdgeOK,
    myMovieStartTime, 0x01000, NULL, &myMovieDuration);

```

Then we can call `DeleteTrackSegment` to delete the track segment occupied by the media sample:

```

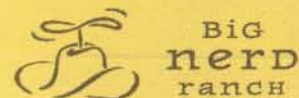
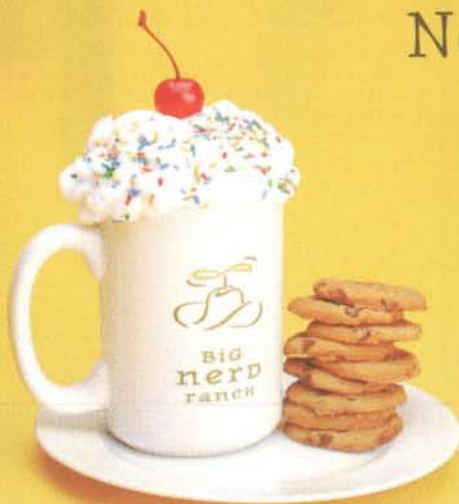
DeleteTrackSegment(myTrack, myMovieStartTime,
    myMovieDuration);

```

Then we proceed as usual, opening a media-editing session and adding the new media sample to the media. The key step here is a call to `AddMediaSample`:

Now serving Cocoa[®] just the way you want it.

Training for Mac OS X doesn't have to be the same old flavor. Reserve your seat in a class at our scenic lodge location, or have experts come to you for **Extreme Mentoring**. Two weeks of on-site instruction and collaboration, customized to the requirements of your project. Book now for 2003. See why we're different.



Intensive Classes for Programmers
www.bignerdranch.com



MACS & FASHION

THE TECHNOLOGY RUNWAY

GLAMOUR, MODELS & MACS > IMAGE BY ANTHONY SAINT JAMES

CREATIVE DESIGNERS, WRITERS, MUSICIANS, BUSINESS LEADERS AND OUR TECHNICAL EXPERT TEAM OFFER THEIR OWN PERSONAL INTERPRETATION OF THINGS THAT ONLY THE MACINTOSH SYSTEM CAN DELIVER. FEATURING OVER 240 PAGES OF REVIEWS, INTERVIEWS PRODUCT NEWS, INSIGHTS, TRENDS AND THE LARGEST MACINTOSH BUYERS' GUIDE. SUBSCRIBE > \$32/1 YEAR (4 ISSUES) OR \$62/2 YEARS (8 ISSUES): WWW.MACDIRECTORY.COM/PAGES/ADVERT.HTML OR SEND CHECK OR MONEY ORDER TO: MACDIRECTORY SUB DEPT. 326 A STREET, 2C, SOUTH BOSTON MA 02110

MacDirectory


```

myErr = AddMediaSample(myMedia,
    mySample,
    0, // no offset in data
    GetHandleSize(mySample),
    myDuration, // frame duration
    (SampleDescriptionHandle)myDesc,
    1, // one sample
    0, // self-contained samples
    &myNewTime);

```

Finally, we insert the media into the track at the desired time and for the desired duration:

```

myErr = InsertMediaIntoTrack(myTrack, myMovieStartTime,
    myNewTime, myDuration, (Fixed)0x00010000L);

```

And we are done! **Listing 10** shows the complete process in one handy routine.

Listing 10: Flattening a child movie into the parent movie file

```

QTMMIM_FlattenChildIntoParent
OSErr QTMMIM_FlattenChildIntoParent
(WindowObject theWindowObject)
{
    Movie myMovie = NULL;
    Track myTrack = NULL;
    Media myMedia = NULL;
    Handle mySample = NULL;
    SampleDescriptionHandle myDesc = NULL;
    TimeValue myMovieTimeNow = 0;
    TimeValue myMediaTimeNow = 0;
    TimeValue myDuration = 0;
    QTAtom myAtom = 0;
    Ptr myDataPtr = NULL;
    Handle myDataRef = NULL;
    long myDataSize = 0;
    OSType myDataRefType;
    Movie myChildMovie = NULL;
    Movie myMemoryMovie = NULL;
    DataReferenceRecord myDataRefRecord;
    Handle myDataRefHandle = NULL;
    Handle myHandleDataRef = NULL;
    Handle myChildDataHandle = NULL;
    Fixed myRate;
    TimeValue myMovieStartTime = 0;
    TimeValue myMovieDuration = 0;
    TimeValue myNewTime = 0;
    OSErr myErr = noErr;

    if (theWindowObject == NULL)
        return(paramErr);

    // round up the usual suspects: the parent movie, movie track, and movie track
    media

    myMovie = (**theWindowObject).fMovie;
    if (myMovie == NULL)
        return(invalidMovie);

    myTrack = GetMovieIndTrackType(myMovie, 1,
        MovieMediaType, movieTrackMediaType);
    if (myTrack == NULL)
        return(invalidTrack);

    myMedia = GetTrackMedia(myTrack);
    if (myMedia == NULL)
        return(invalidMedia);

    // get the child movie sample data

    // if the parent movie is playing, stop it
    myRate = GetMovieRate(myMovie);
    SetMovieRate(myMovie, 0);

    mySample = NewHandleClear(0);
    if (mySample == NULL)
        return(MemError());

```

```

myDesc = (SampleDescriptionHandle)NewHandleClear(0);
if (myDesc == NULL) {
    myErr = MemError();
    goto bail;
}

myMovieTimeNow = GetMovieTime(myMovie, NULL);
if (myMovieTimeNow == GetMovieDuration(myMovie))
    myMovieTimeNow--;

myMediaTimeNow = TrackTimeToMediaTime(myMovieTimeNow,
    myTrack);
if (myMediaTimeNow == -1) {
    myErr = invalidTime;
    goto bail;
}

myErr = GetMediaSample(myMedia, mySample, 0, NULL,
    myMediaTimeNow, NULL, &myDuration, myDesc, NULL, 1,
    NULL, NULL);
if (myErr != noErr)
    goto bail;

// the media sample is an atom container;
// find the data reference atom inside the media sample

myAtom = QTFindChildByIndex(mySample,
    kParentAtomIsContainer, kMovieMediaDataReference,
    1, NULL);
if (myAtom != 0) {
    // get the data reference atom data
    QTLockContainer(mySample);

    myErr = QTGetAtomDataPtr(mySample, myAtom, &myDataSize,
        &myDataPtr);
    if (myErr != noErr)
        goto bail;

    myDataRefType = EndianU32_BtoN(*(OSType *)myDataPtr);
    myErr = PtrToHand(myDataPtr + sizeof(OSType),
        &myDataRef, myDataSize - sizeof(OSType));
    if (myErr != noErr)
        goto bail;

    QTUnlockContainer(mySample);

    // open the child movie and flatten it entirely into memory

    myErr = NewMovieFromDataRef(&myChildMovie, 0, NULL,
        myDataRef, myDataRefType);
    if (myErr != noErr)
        goto bail;

    myDataRefHandle = NewHandleClear(0);
    if (myDataRefHandle == NULL)
        goto bail;

    myHandleDataRef =
        QTDR_MakeHandleDataRef(myDataRefHandle);
    if (myHandleDataRef == NULL)
        goto bail;

    myDataRefRecord.dataRefType = HandleDataHandlerSubType;
    myDataRefRecord.dataRef = myHandleDataRef;

    myMemoryMovie = FlattenMovieData(myChildMovie,
        flattenFSSpecPtrIsDataRefRecordPtr |
        flattenAddMovieToDataFork,
        (FSSpecPtr)&myDataRefRecord,
        sigMoviePlayer,
        smSystemScript,
        0L);
    myErr = GetMoviesError();
    if (myErr != noErr)
        goto bail;

    DisposeMovie(myChildMovie);

    // replace the existing data reference atom by a "flattened" data reference atom

    // out with the old...
    QTRemoveAtom(mySample, myAtom);

```


NOW RUNNING ON YOUR MAC!

```
// ...and in with the new
myChildDataHandle = NewHandleClear(sizeof(OSType) +
    sizeof(Handle));
if (myChildDataHandle != NULL) {
    OSType myType;

    // set the data reference type
    myType = EndianU32_NtoB(HandleDataHandlerSubType);
    BlockMove(&myType, *myChildDataHandle,
        sizeof(OSType));

    // leave the next four bytes set to 0x00000000;
    // add a filenaming extension and an initialization extension
    myErr = QTDR_AddFilenamExtension
        (myChildDataHandle, NULL);
    if (myErr != noErr)
        goto bail;

    HLock(myDataRefHandle);
    myErr = QTDR_AddInitDataDataRefExtension
        (myChildDataHandle, *myDataRefHandle);
    HUnlock(myDataRefHandle);
    if (myErr != noErr)
        goto bail;

    HLock(myChildDataHandle);
    myErr = QTInsertChild(mySample,
        kParentAtomIsContainer, kMovieMediaDataReference,
        1, 1, GetHandleSize(myChildDataHandle),
        *myChildDataHandle, NULL);
    HUnlock(myChildDataHandle);
    if (myErr != noErr)
        goto bail;
}

// add the new sample to the media

// determine the bounds of this sample in movie time
GetTrackNextInterestingTime(myTrack,
    nextTimeMediaSample | nextTimeEdgeOK,
    myMovieTimeNow, -0x01000, &myMovieStartTime, NULL);
GetTrackNextInterestingTime(myTrack,
    nextTimeMediaSample | nextTimeEdgeOK,
    myMovieStartTime, 0x01000, NULL, &myMovieDuration);

// splice this media over the old one
DeleteTrackSegment(myTrack, myMovieStartTime,
    myMovieDuration);

myErr = BeginMediaEdits(myMedia);
if (myErr != noErr)
    goto bail;

// write a new media sample into the track
myErr = AddMediaSample(myMedia,
    mySample,
    0, // no offset in data
    GetHandleSize(mySample),
    myDuration, // frame duration
    (SampleDescriptionHandle)myDesc,
    1, // one sample
    0, // self-contained samples
    &myNewTime);

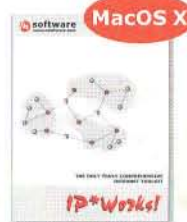
myErr = EndMediaEdits(myMedia);
if (myErr != noErr)
    goto bail;

// add the media to the track
myErr = InsertMediaIntoTrack(myTrack, myMovieStartTime,
    myNewTime, myDuration, (Fixed)0x00010000L);

} else {
    myErr = cannotFindAtomErr;
}

bail:
if (mySample != NULL)
    DisposeHandle(mySample);

if (myDesc != NULL)
    DisposeHandle((Handle)myDesc);
```



IP*Works! The Only Truly Comprehensive Internet Toolkit

- More Than 30 Components Cover All Major Internet Protocols
- Follows Exact RFC Specifications
- Market-Tested For Over 7 Years
- In Use By Almost All Fortune 500s
- Royalty-Free Pricing

IP*Works! for Mac OS X (10.0/10.1/10.2) gives you access to a host of new capabilities that will connect your applications to the Internet with unprecedented ease of use, power, and flexibility! You will be able to easily and quickly:

- program the web: HTTP, WebForm, WebUpload
- call web services: SOAP, XMLp
- transfer files: FTP, TFTP
- send email: SMTP, FileMailer, HTMLMailer
- receive email: POP, IMAP
- read/post news: NNTP
- encode/decode: MIME, NetCode
- access directories: LDAP
- manage networks: SNMP, Whois, Ping, TraceRoute
- build clients and servers: IPPort, IPDaemon
- build packet applications: UDPPort, Multicast
- remote access: Telnet, Rexec, Rshell, RCP

...and a whole lot more! - download your free trial today from www.nsoftware.com!

IP*Works! for MacOS X is currently available as a C/C++ Library (OS X Framework). Objective-C Classes for Cocoa and RealBasic plugins coming soon at www.nsoftware.com

STAY TUNED!


```

if (myDataRef != NULL)
    DisposeHandle(myDataRef);

if (myDataRefHandle != NULL)
    DisposeHandle(myDataRefHandle);

if (myHandleDataRef != NULL)
    DisposeHandle(myHandleDataRef);

if (myChildDataHandle != NULL)
    DisposeHandle(myChildDataHandle);

// restore the original movie rate
SetMovieRate(myMovie, myRate);

return(myErr);
}

```

As written, `QTMIM_FlattenChildIntoParent` replaces the current movie media sample by a “flattened” sample. It would be easy to adapt this routine to iterate through all samples in the movie track and to flatten each child movie into the parent movie. I’ll leave this refinement as an exercise for the reader.

MEMORY-BASED MOVIES

Let’s finish up this article by taking a look at a few ways we can create movies or tracks whose media data is contained entirely in memory. This is a useful thing to do for a number of reasons. For example, we might generate all of a movie’s media data dynamically and not want to have to create a disk file to hold that data. Or we might want to add a track to an existing movie but don’t want the track’s media data to be added to the associated movie file unless the user explicitly requests it. In that case, we can tell the track to store its media data in memory, not in the original movie file.

The key element here is to create a handle data reference and to set it as the data reference for the movie (or track). Any media data written to the movie (or track) will be written into memory, at the location specified by the data reference.

Creating Movies in Memory

In a previous article (“Somewhere I’ll Find You”, cited earlier), we saw how to create a movie whose associated media data is contained entirely in RAM. There, we took advantage of the fact that `FlattenMovieData` can flatten a movie into a location specified by a data reference instead of by a file system specification record. **Listing 11** shows the core of our code for doing this. (This should remind you of **Listing 7** above.)

Listing 11: Flattening a movie into memory

```

Movie          myNewMovie = NULL;
Handle         myDataRef = NULL;
Handle         myHandle = NULL;
DataReferenceRecord myDataRefRecord;

myHandle = NewHandleClear(0);
if (myHandle == NULL)
    goto bail;

myDataRef = QTDR_MakeHandleDataRef(myHandle);
if (myDataRef == NULL)
    goto bail;

myDataRefRecord.dataRefType = HandleDataHandlerSubType;

```

```

myDataRefRecord.dataRef = myDataRef;

myNewMovie = FlattenMovieData(myMovie,
                               flattenFSSpecPtrIsDataRefRecordPtr,
                               (FSSpecPtr)&myDataRefRecord,
                               sigMoviePlayer,
                               smSystemScript,
                               0L);

```

Using `FlattenMovieData` assumes that we already have a movie (`myMovie`) and want to copy it entirely into RAM. It’s sometimes also useful to create a new movie in RAM, using the `NewMovie` function and our standard calls to `NewMovieTrack`, `NewTrackMedia`, and so forth. The easiest way to do this is to set the movie’s *default data reference* to a handle data reference, using the `SetMovieDefaultDataRef` function. We can call `SetMovieDefaultDataRef` like this:

```

myErr = SetMovieDefaultDataRef(myMovie, myDataRef,
                               HandleDataHandlerSubType);

```

Listing 12 defines a function, `QTDR_CreateMovieInRAM`, that creates a new movie whose media data is stored in a block of memory.

Listing 12: Creating a movie in memory

```

Movie QTDR_CreateMovieInRAM (void)
{
    Movie          myMovie = NULL;
    Track          myTrack = NULL;
    Media          myMedia = NULL;
    short          myResRefNum = 0;
    short          myResID = 0;
    Handle         myDataRef = NULL;
    Handle         myHandle = NULL;
    FSSpec         myFSSpec;
    OSErr          myErr = noErr;

    // create a new handle to hold the media data
    myHandle = NewHandleClear(0);
    if (myHandle == NULL)
        goto bail;

    // create a data reference to that handle
    myDataRef = QTDR_MakeHandleDataRef(myHandle);
    if (myDataRef == NULL)
        goto bail;

    myMovie = NewMovie(newMovieActive);
    if (myMovie == NULL)
        goto bail;

    myErr = SetMovieDefaultDataRef(myMovie, myDataRef,
                                   HandleDataHandlerSubType);
    if (myErr != noErr)
        goto bail;

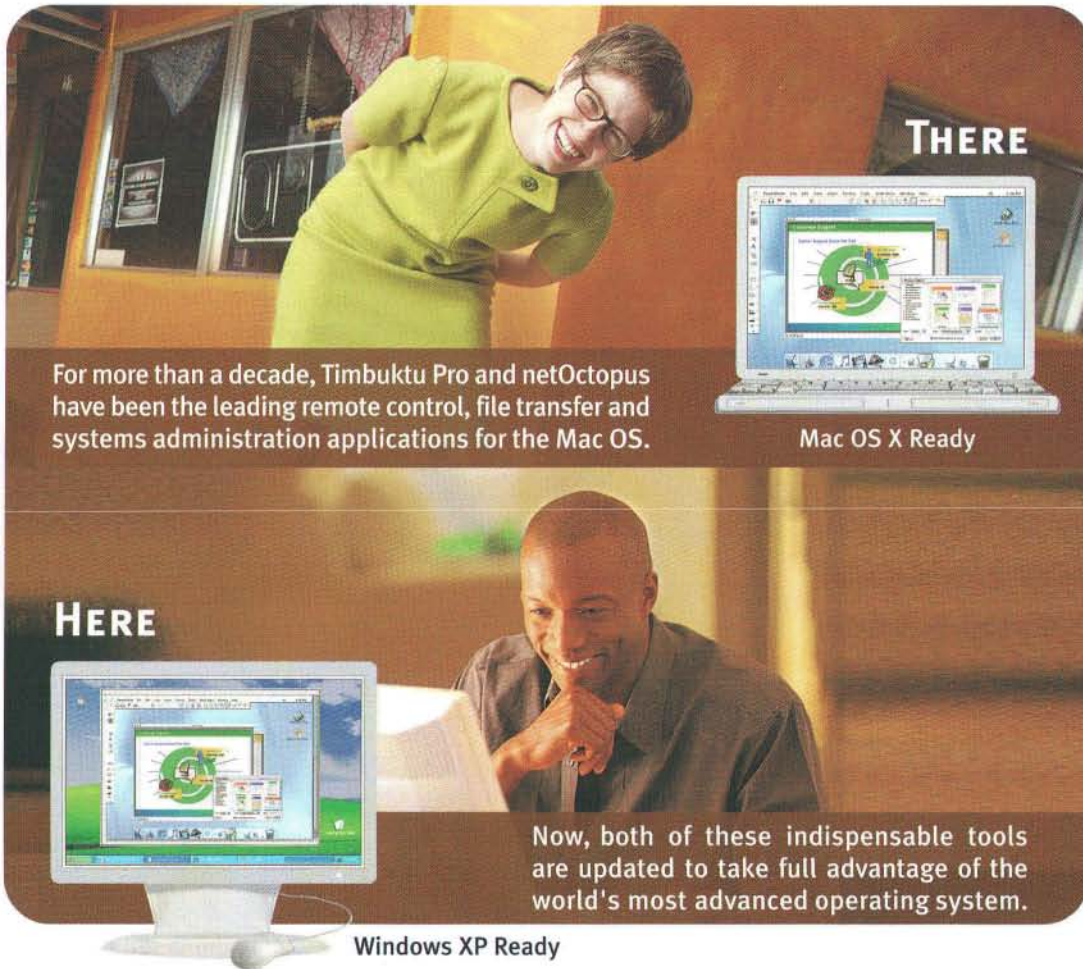
    // create the movie track and media
    myTrack = NewMovieTrack(myMovie,
                           FixRatio(kVideoTrackWidth, 1),
                           FixRatio(kVideoTrackHeight, 1), kNoVolume);
    myErr = GetMoviesError();
    if (myErr != noErr)
        goto bail;

    myMedia = NewTrackMedia(myTrack, VideoMediaType,
                           kVideoTimeScale, NULL, 0);
    myErr = GetMoviesError();
    if (myErr != noErr)
        goto bail;

    // create the media samples
    myErr = BeginMediaEdits(myMedia);
    if (myErr != noErr)
        goto bail;
}

```


Stay In Control Wherever You Go.



THERE

For more than a decade, Timbuku Pro and netOctopus have been the leading remote control, file transfer and systems administration applications for the Mac OS.

Mac OS X Ready

HERE

Now, both of these indispensable tools are updated to take full advantage of the world's most advanced operating system.

Windows XP Ready

Timbuku Pro

Whether you're at home or at work, Timbuku Pro allows you to operate distant computers as if you were sitting in front of them, transfer files or folders quickly and easily, and communicate by instant message, text chat, or voice intercom.

<http://www.timbukupro.com>

netOctopus

Intuitive and powerful, netOctopus can manage a network of ten or 10,000 computers. Inventory computers, software and devices on your network; distribute software; configure remote computers; and create custom reports on the fly.

<http://www.netoctopus.com>

Learn more, try it, or buy it online. Call us at **1-800-485-5741**.



timbuku® • netOctopus®

netopia®


```

myErr = QTDR_AddVideoSamplesToMedia(myMedia,
    kVideoTrackWidth, kVideoTrackHeight);
if (myErr != noErr)
    goto bail;

myErr = EndMediaEdits(myMedia);
if (myErr != noErr)
    goto bail;

// add the media to the track
myErr = InsertMediaIntoTrack(myTrack, 0, 0,
    GetMediaDuration(myMedia), fixed1);
if (myErr != noErr)
    goto bail;

// add the movie atom to the movie file
AddMovieResource(myMovie, myResRefNum, &myResID, NULL);

bail:
if (myDataRef != NULL)
    DisposeHandle(myDataRef);

return(myMovie);
}

```

This function is virtually identical to other movie-creating functions we've seen in previous articles, except that it calls `SetMovieDefaultDataRef` to cause all media data to be written to a block of memory. Previously, we relied on the fact that a new movie's default data reference is the file opened by a call to `CreateMovieFile` or `NewMovieFromFile`. Here we are calling `NewMovie` to create a new movie with no attachment to any existing file, so we need to explicitly set the movie's default data reference.

If we want just a particular track to have its media data in memory, then we can pass a handle data reference when calling `NewTrackMedia`, like this:

```

myMedia = NewTrackMedia(myTrack, VideoMediaType,
    myTimeScale, myDataRef, HandleDataHandlerSubType);

```

The specified data reference overrides the default movie data reference.

Saving Movies from Memory

In all these cases, we've ended up with a movie that has some or all of its media data stored directly in memory, accessed using a handle data reference. We can, of course, play the movie, edit the movie, enable and disable tracks (and so forth), exactly as if the media data were contained in a file accessed via a file data reference or stored remotely and accessed via a URL data reference. But what happens if we want to save this movie into a file on disk? Well, it depends. If we created the movie entirely in memory (as in Listings 11 and 12), then our underlying application framework code will detect that no file is attached to the movie yet. In this case, it will elicit a filename from the user, create a new file (or delete and recreate an existing file), and then call `FlattenMovieData` to write the media data into the new file. `FlattenMovieData` strips out any unneeded media samples, resolves all remaining media sample references, and writes the media data into the movie file on disk. This movie

FREE UPGRADES FOR LIFE!

TIRED OF BEING SHAKEN DOWN FOR EXPENSIVE, BUGGY, FORCED UPGRADES FROM THE BIG VENDORS? JOIN THE PEOPLE WHO BELIEVE THAT YOU SHOULD BUY SOMETHING ONCE – AND HAVE IT FOREVER! SUPPORT AN OPEN AND FREE SOFTWARE DEVELOPMENT COMMUNITY NOT RUN BY MARKETING TYPES OR BEAN COUNTERS.



The applications in Stone Studio — starring Create® — have all the most used features of these applications: Illustrator, Quark Xpress, DreamWeaver, Freehand, Corel Draw, PageMaker, Timeslips, Acrobat Writer, Stuffit Deluxe, ImageReady, iPhoto and more. Visit www.stone.com/Stone_Studio_Successes to see how people just like you get their best ideas out into the world.



STONE STUDIO™

7 APPS 1 LOW PRICE 0 PAID UPGRADES

STONE STUDIO DOWNLOAD NOW FROM WWW.STONE.COM



\$299

The Key is in Your Hands!

Does your dongle do all this?

Software Goes Online

Electronic Software Distribution – safely protected by WIBU-KEY.

Web Remote Programming

Reprogram the WIBU-BOX hardware at the customer's site directly via the Internet.

Web Authentication

Secure authentication via a two-way-encryption.

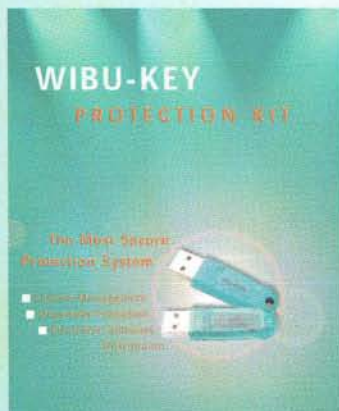
Pay-Per-Use

Usage dependent accounting.

Mac OS 9 & X

WIBU-KEY supports Mac OS, Windows and heterogeneous networks.

► **Yes? Then you are already using WIBU-KEY!**



► **No? Then order your Test Kit
at 1-800-986-6578**

**You will find more information at
www.griftech.com**

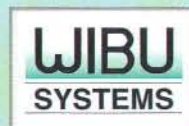


USA, Canada: Griffin Technologies, LLC
phone: (785) 832-2070 · fax: (785) 832-8787
email: sales@griftech.com · www.griftech.com

or visit our booth at...



...and see our products



WIBU-SYSTEMS AG
76137 Karlsruhe, Germany
WIBU-SYSTEMS USA, Inc.
Seattle, WA 98101
email: info@wibu.com

www.wibu.com

Test Kits also available at:

Argentina info@safied-group.com, Australia simone.ecklerle@wibu.com, Belgium wibu@impakt.be, Croatia aries@aries.hr, Denmark lean@danbit.dk, Egypt esafwat@odec.com.eg, Finland finebyte@finebyte.com, France info@neol.fr, Hungary info@mrsoft.hu, Japan info@suncarla.co.jp, Jordan, Lebanon starsoft@cyberia.net.lb, Korea dhkimm@wibu.co.kr, Luxembourg wibu@impakt.be, Netherlands wibu@impakt.be, Portugal dubit@dubit.pt, Syria starsoft@cyberia.net.lb, Thailand preecha@dpf-th.com, United Kingdom info@codework.com, USA sales@griftech.com

file contains only file data references. It's a self-contained movie file that holds all of its media data. So far, so good.

Things start to get interesting, however, if we've already got a movie file attached to our movie. (This would happen, for instance, if we open an existing movie file and then add a track whose media data is accessed using a handle data reference.) In this case, when the user decides to save the movie, our application framework code calls `UpdateMovieResource` instead of `FlattenMovieData`. `UpdateMovieResource` does not write any media data into the movie file; rather, it simply updates the movie atom, which contains the data references for each media. The updated movie file now contains a handle data reference. The problem here is that, when the movie file is closed and then reopened, QuickTime won't be able to find the media data. The handle data reference, in all likelihood, no longer picks out any valid media data.

Certainly one way to avoid this problem is to make sure that we call `FlattenMovieData` at least once before we close a movie file. But that might not be desirable in some instances. For example, our movie might contain references to other files, in addition to the references to memory-based data. We might not want to force *all* data references to be resolved, just one or two of them.

As far as I know, QuickTime doesn't currently provide a way to flatten only selected tracks in a movie. We can work around this limitation, to some degree, by employing a simple technique involving initialization extensions. The idea is to "smuggle" a track's media data into the movie atom, by attaching that data to the media's data reference as an initialization extension. When we then call `UpdateMovieResource`, the media data will be written to the movie file, since it now forms part of the movie atom. The movie file once again contains a handle data reference, but it's harmless; when QuickTime reopens the movie file, it will notice the data reference extension and load the media data from that extension. Sweet.

It's actually quite easy to implement this smuggling. We do it by passing a handle data reference to `NewTrackMedia`, just as we did at the end of the previous section. But this time, instead of passing a handle data reference for a handle to a 0-length block of data, we'll pass a handle data reference for a `NULL` handle, where the handle data reference has an initialization extension. Let's begin by creating a handle data reference:

```
myDataRef = NewHandleClear(sizeof(Handle) + sizeof(char));
```

Remember that a handle data reference is a handle to a handle. Here we've created a handle to a 5-byte block of memory, all of whose bytes are set to 0. This represents a `NULL` handle and a 0-length filename extension. At this point, we'll tack on the atom header for the initialization extension:

```
myAtomHeader[0] = EndianU32_NtoB(sizeof(myAtomHeader));
myAtomHeader[1] = EndianU32_NtoB(
```

```
kDataRefExtensionInitializationData);
myErr = PtrAndHand(myAtomHeader, myDataRef,
    sizeof(myAtomHeader));
```

We haven't actually added any initialization data to the data reference extension, only the 8-byte atom header. But that's all we need at this point. We're ready to call `NewTrackMedia`:

```
myMedia = NewTrackMedia(myTrack, VideoMediaType,
    kVideoTimeScale, myDataRef,
    HandleDataHandlerSubType);
```

When QuickTime sees the initialization extension atom header in the handle data reference, it knows to keep the media data in the data reference itself, rather than in the memory block addressed by the handle that forms the first four bytes of the data reference's referring data. The handle data handler is going to allocate whatever memory is needed to hold the data we add to our media, so we can dispose of our data reference (`myDataRef`) immediately if we like.

Now we can edit the media and track as usual, for instance by calling `AddMediaSample` and `InsertMediaIntoTrack`. When we subsequently call `UpdateMovieResource`, the handle data handler will write out a handle data reference with a fully-configured initialization extension.

Listing 13 shows most of this assembled into a single routine, `QTDR_CreateTrackInRAM`. It adds a new video track to a movie, with the track's media data stored in RAM. In addition, the media data will be written into the data reference as an initialization extension when the movie atom is updated.

Listing 13: Creating a track in memory

```
QTDR_CreateTrackInRAM
OSError QTDR_CreateTrackInRAM (Movie theMovie)
{
    Track          myTrack = NULL;
    Media          myMedia = NULL;
    Handle         myDataRef = NULL;
    unsigned long  myAtomHeader[2];
    OSError        myErr = noErr;

    if (theMovie == NULL)
        return(paramErr);

    myDataRef = NewHandleClear(sizeof(Handle) +
                               sizeof(char));

    if (myDataRef == NULL)
        return(MemError());

    myAtomHeader[0] = EndianU32_NtoB(sizeof(myAtomHeader));
    myAtomHeader[1] = EndianU32_NtoB(
        kDataRefExtensionInitializationData);

    myErr = PtrAndHand(myAtomHeader, myDataRef,
        sizeof(myAtomHeader));
    if (myErr != noErr)
        goto bail;

    // create the movie track and media
    myTrack = NewMovieTrack(theMovie,
        FixRatio(kVideoTrackWidth, 1),
        FixRatio(kVideoTrackHeight, 1), kNoVolume);
    myErr = GetMoviesError();
    if (myErr != noErr)
        goto bail;

    myMedia = NewTrackMedia(myTrack, VideoMediaType,
        kVideoTimeScale, myDataRef,
        HandleDataHandlerSubType);
```



```

myErr = GetMoviesError();
if (myErr != noErr)
    goto bail;

// create the media samples
myErr = BeginMediaEdits(myMedia);
if (myErr != noErr)
    goto bail;

myErr = QTDR_AddVideoSamplesToMedia(myMedia,
    kVideoTrackWidth, kVideoTrackHeight);
if (myErr != noErr)
    goto bail;

myErr = EndMediaEdits(myMedia);
if (myErr != noErr)
    goto bail;

// add the media to the track
myErr = InsertMediaIntoTrack(myTrack, 0, 0,
    GetMediaDuration(myMedia), fixed1);

bail:
if (myDataRef != NULL)
    DisposeHandle(myDataRef);

return(myErr);
}

```

This is a neat technique, but it's got a few limitations that you should know about. First of all, it works only with QuickTime 4.0 and later. Under earlier versions of QuickTime, data reference extensions are simply ignored. Also, and more importantly, because the initialization extension is stored inside the movie atom, you should avoid creating very large extensions. The extensions will remain in RAM for significant periods of time, so it's good to keep them small.

CONCLUSION

Data. We've gotta have it, at least if we want to do anything very interesting in our QuickTime movies. In this article, we've taken a look at a couple of useful ways of managing a movie's media data. First, we saw how to construct a movie that picks out its media data using media sample references. These references can refer to existing data that lives outside the movie file (as in the case of our slide show movie) or that is already contained in the movie file (as in the case of our effects movie). A sample reference is simply a way to make use of some existing data without having to copy it into a movie file or between tracks.

We've also seen, however, that it's sometimes useful to be able to go in the reverse direction, by forcing a movie's media data to be packed into an existing movie file. (At the very least, this makes it much easier to move the movie file around, since we don't need to worry about moving any other files that the movie depends upon.) Our standard means of doing this is to call `FlattenMovieData`, but sometimes that either doesn't work at all (as in the case of child movies) or doesn't work selectively enough for our purposes (as in the case of a single memory-based track). To work around some of the limitations of `FlattenMovieData`, we can use initialization data data reference extensions to attach media data directly to a data reference.

THE #1 RELATIONAL DATABASE ON MAC OS X

OPENBASE SQL

HIGH-PERFORMANCE RELATIONAL DATABASE



THE DATABASE THAT PAYS YOU CASH

*Announcing unlimited single-user
runtimes for just \$350 a year.*

Distribute your software with a royalty-free single-user version of OpenBase SQL. Receive monthly revenue when your users upgrade to one of our low-cost multi-user subscription licenses.

Go to www.openbase.com/cashback for more details!

Visit our Web site and check out the innovative features that make OpenBase SQL the database of choice for developers.

www.openbase.com



CREATE WITH POWER™

by Dave Mark

Getting Started: Circa 2002

I have to tell you, it feels great to be back. It's been a little more than five years since my last Getting Started column, but it feels like an eternity. The long and winding road from the days of *Inside Macintosh*, Nubus cards, black and white monitors, GetPort() and SetPort(), trap patching, 68K emulation, etc. has turned a radical new corner.

Though the vast majority of all Macs still run System 9 or earlier, the future is clear. This is a unique moment in time and you have a unique opportunity. Back in the frontier days of Macintosh development, an unusual mix of elements was taking shape. There was a wonderful new computing platform that freed you from the traditional bonds of DOS. Instead of a limited (and frequently cryptic) set of commands that governed your interaction with your computer, applications such as MacWrite and MacPaint allowed you to express yourself in revolutionary new ways.

As with any revolutionary change, opportunities blossomed. People loved the Macintosh and, as word spread, their hunger for new applications grew dramatically. Problem was, developing a Mac application was completely different than the relatively simple process of building a DOS or Unix app. Back in the day, *Inside Macintosh* was a single volume that came in a loose leaf binder, and deciphering its mysteries required dedication and a great deal of trial and error. But, for those "in the know", there was money to be made. Startups were everywhere. This was fertile ground. Exciting times.

I loved those early days. The excitement of learning about something so new and so beautifully crafted inspired me. And once I understood the basics, I felt compelled to share my knowledge with the Mac development community. I wrote books like the *Macintosh Programming Primer* series, *Learn C on the Macintosh* and, of course, 7 years worth of *Getting Started* columns for MacTech.

Over time, each new release of the Mac OS brought less dramatic changes and the process moved from revolutionary to evolutionary. As the process of building a Mac application became less mystical and more practical, money came to the table and getting a Mac application to market and competing with established brands became harder and much more

expensive. What was once a fun, "programming for the beauty of it" process moved from the computer science end of the spectrum to the marketing end of the spectrum. The pioneer days were dead.

And now we've come full circle. Mac OS X is a whole new beast. New APIs to learn, a new development environment to explore, new widgets to play around with. We've got a whole new frontier to explore and there are opportunities here for all of us.

GETTING STARTED

There are a number of paths to explore here. We'll start with the most straightforward, building an app using the Objective C programming language and Apple's Cocoa framework. Over the coming months, we'll dig into the basics of Objective C. Over time, we'll take on Cocoa and, eventually, explore some of the other paths to build our Mac OS X apps. I'd also like to spend some time under the hood, exploring the OS upon which Mac OS X is based, Apple's port of the Unix operating system.

Go get the tools

There are a number of good choices out there when it comes to development tools. There's CodeWarrior from Metrowerks, REALbasic from REAL Software, a variety of AppleScript environments, and many others. For the moment, we're going to focus our attention on the tools that Apple provides, gratis, to ensure we all start on the same footing.

When Apple bought NeXT back in December of '96, they got Steve Jobs back and they also got Steve's OS and the development tools crafted to work alongside the OS. The OS evolved into Mac OS X, and the tools became Apple's official Mac OS X dev tools.

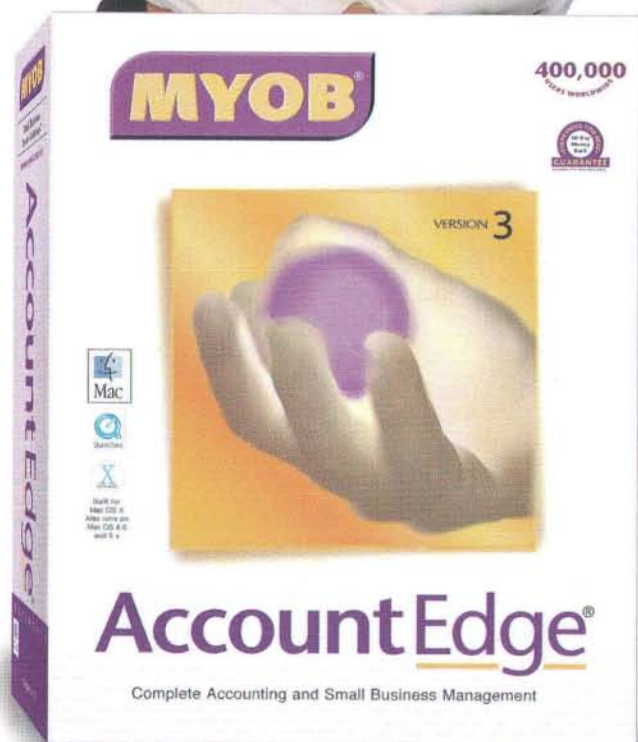
Many of you already have Apple's dev tools in hand. Newer Macs ship with a Developer Tools CD. You may find that your Mac came with the tools pre-installed. Nonetheless, go through the web site, download the latest and greatest, and install them on your machine.

Dave Mark is very old. He's been hanging around with Apple since before there was electricity and has written a number of books on Macintosh development, including *Learn C on the Macintosh*, *Learn C++ on the Macintosh*, and *The Macintosh Programming Primer* series. Dave maintains a primitive web site at <http://www.spiderworks.com>

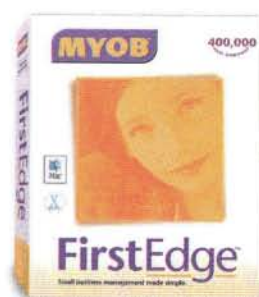
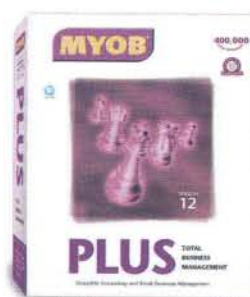
CONFIDENCE



is knowing
you've got
what it takes!



MYOB
what it takes
to
manage your small business
successfully.



Award Winning Small Business Accounting and Management Software

Join the over 400,000 MYOB customers worldwide who trust their business success to MYOB products. From business basics to complete management, MYOB has what it takes to Mind Your Own Business better. Available for Macintosh and Windows.

Visit us online for a **FREE Trial Version**

www.myob.com/us/mt

800-322-MYOB(6962)



**Small Business.
Smart Solutions.™**

First step: Navigate to:

<http://developer.apple.com>

This is the home for the Apple Developer Connection, also known as ADC. There is a ton of great material on this site. You can sign up for Apple's various developer programs, including the Premier (US\$3500 per year), Select (US\$500 per year), Student (US\$99 per year), Mailing (US\$199 per year), and Online (FREE) programs. Take some time to go through the program descriptions to see if one of them is right for you.

To get the tools, all you have to do is register for the Online program. To register, send your browser to:

<http://connect.apple.com>

Click the "Join" button, read the license, click "Agree", then fill out the form and select your new Apple ID. Once your account is set up, log in, then select Download Software from the nav bar on the left hand side of the ADC window. Next, click Mac OS X from the sub-nav bar (**Figure 1**).

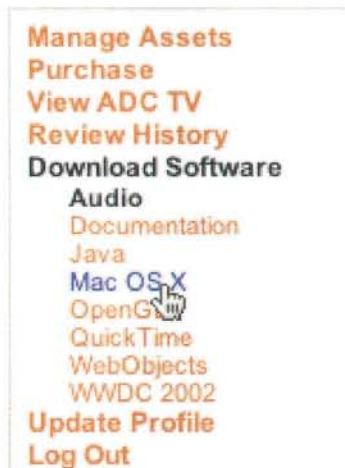


Figure 1. Click the Download Software link, then Mac OS X

There are a lot of choices on this page. As I write this, the latest releases are the *July 2002 Mac OS X 10.2 Developer Tools* and the *August 2002 Dev Tools 10.2 Update*. By the time this column gets to you, however, there may be a new release of the tools or another update. As a rule, download the most recent *Developer Tools* package first. Then, check to see if an *Update* package was released after the *Developer Tools* package. If so, download it as well. Install the *Developer Tools* package and the *Update*, if applicable.

Checking the Install

Once the tools are installed, you should have a directory named Developer at the top level of your hard drive. My Developer directory listing is shown in **Figure 2**. Take a few moments to go through the various Developer sub-directories.

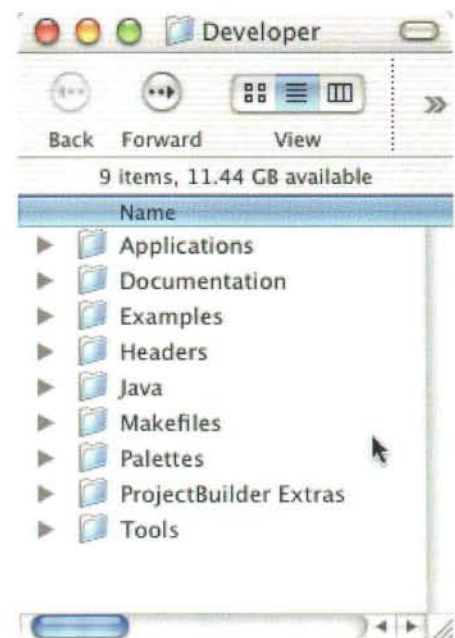


Figure 2. The Developer directory after the tools install.

In the Applications directory, you'll find a number of interesting tools, including one we'll fire up in just a sec called Project Builder. For you CodeWarrior fans, Project Builder is sort of the equivalent of the CodeWarrior IDE, a command central for all your project files and the application that actually calls the compiler and other code building tools.

The Documentation folder is also filled with important goodies. Get to know what docs are available. There's a lot of good reading in there, and it's all free. One short file worth reading is the README.html file in the Documentation directory. This file will open in your web browser and lists various ways to access the documentation from within Project Builder.

If you'd like to get ahead of the game, take a look at the file:

`/Developer/Documentation/Cocoa/ObjectiveC/ObjC.pdf`

ObjC.pdf will give you a fairly thorough grounding in the Objective C language.

Take 'em For a Spin

Now that the tools are installed, let's take them for a spin. This month's project will be a simple C "Hello World" project, just to get a sense of the environment. Next month, we'll try our hand at some Objective C code.

Navigate into the `/Developer/Applications` folder and launch Project Builder. Select New Project... from the File menu. The New Project dialog appears, allowing you to specify the type of project you'd like to build. Scroll all the way to the bottom and select Standard Tool (**Figure 3**). Standard Tool builds an ANSI C command line program. Click the Next button.

Ever consider exhibiting at MacWorld?
Thought it might break your budget?

THINK AGAIN!

The most affordable solutions. Completely turnkey.

9 Pavilions to Choose From:

- Bluetooth & Wireless Technologies
- Enterprise, Networking & Server Solutions
- Education, Edutainment & Assistive Technologies
- MacTech Central
- Digital Media
- QuickTime
- International
- SciTech Pavilion
- Business Solutions

LIMITED TIME SPECIAL DISCOUNT OFFER:
Get \$1000 off your station package!

If you sign up by October 31, 2002, you'll get US\$1000 off your station, bringing the price to just US\$2995!

Send e-mail or visit the pavillion web site for more details.

reservations@xplain.com

<http://www.xplain.com/macworldexpo>

805/494-9797

Xplain Corporation • PO Box 5200 • 850-P Hampshire Road • Westlake Village, CA 91359-5200
Voice: 805-494-9797 • Fax: 805-494-9798 • reservations@xplain.com

Conferences January 6 – 10, 2003

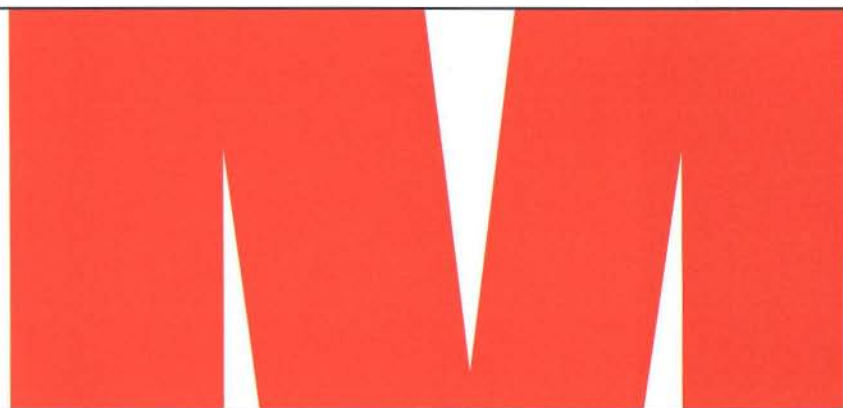
Expo January 7 – 10, 2003

San Francisco The Moscone Center

www.macworldexpo.com

January 6 – 10

Macworld
Conference & Expo.



Macworld Conference & Expo is recognized as the "must-attend" event for the Mac community. Join tens-of-thousands of attendees this January to benefit from high-level education and enjoy the strong community and camaraderie that takes place at this all-in-one marketplace.

"The best single source for information on the Mac. Even Windows users would be enlightened."

Ken, President

"If you use a Mac, you are not using it to its fullest potential until after you attend Macworld."

Michael, Graphic Artist



Flagship Sponsors

Macworld

Macworld.com



MacCentral

Platinum Sponsor

COREL

SF03

www.macworldexpo.com



Acquire what you need (knowledge, products, services or solutions) to stay competitive and on the cusp of technology.

Enjoy the one-stop-shop atmosphere only Macworld Conference & Expo can provide you.

Test drive brand new products and services — be one of the first to kick the tires of the latest innovations.

Apply knowledge learned from 5 intense educational days at Macworld Conference & Expo immediately.

Network, exchange ideas and build contacts with like-minded, or not so like-minded, users and industry gurus.

Feel what it is like to be part of a loyal, powerful and holistic community.

Discuss your issues, mention your concerns, or praise the manufacturers of your favorite products directly.

Register online with Priority Code: A-MTN

For more information, call toll free 1-800-645-EXPO

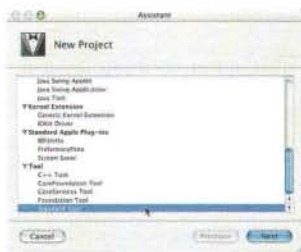


Figure 3. Select the Standard Tool project template.

You'll be prompted for a directory in which to store all the project related files (**Figure 4**), including all source, object, and binaries. Name your project Hello World, then click the Choose... button to browse on your hard drive for a location for the Hello World folder. I created a Projects folder within my personal folder rather than storing the projects within the /Developer directory. I don't want to wipe out my projects when I decide to do a wipe and reinstall of the dev tools.

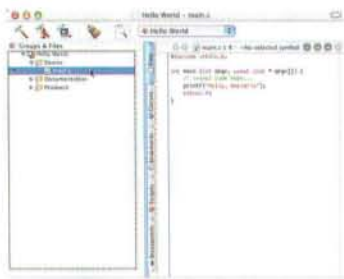


Figure 4. Saving your new project files.

Now click the Finish button. Project Builder will build a project for you, complete with a source code file named main.c containing a main() function any C programmer will recognize in a heartbeat.

The project window that appears contains a number of elements. We'll get into them in detail in next month's column. For now, the important elements are the "Groups & Files" pane, the code editing pane, and the toolbar (at the top of the window, the one with all the funny hammer icons).

In the "Groups & Files" pane, click on the disclosure triangle to the left of the Source folder icon. You'll reveal a single file named main.c within the Source group. Click on main.c. Notice the source code that appears in the code editing pane (**Figure 5**).

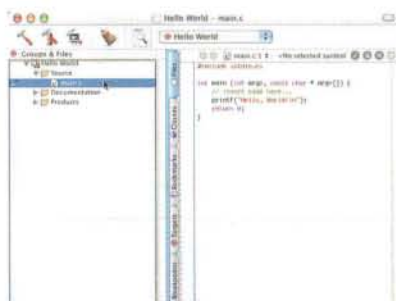


Figure 5. The Hello World source code listed in the project window's code editing pane.

Let's run this sucker. Click on the 3rd icon in the toolbar (the one with the hammer covered by a computer display). If you hover over the icon, a tooltip appears with the words "Build and run active executable". That's the one we want. This will compile our source, link our object code into an executable and run the executable. Do it.

Your result should be eerily similar to the one shown in **Figure 6**. Cool!

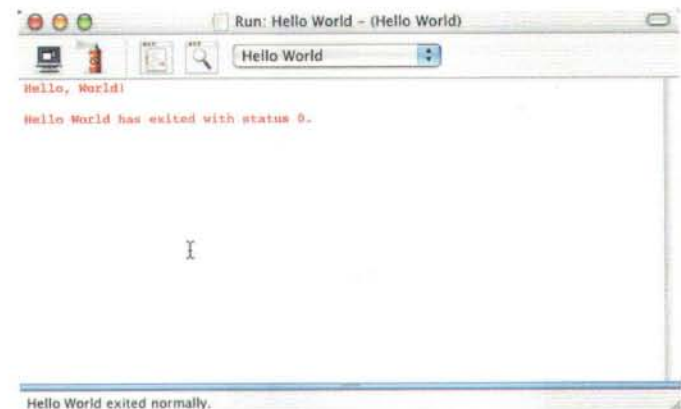


Figure 6. Hello World does its thing.

When you asked Project Builder to build and run your project, Project Builder did just that. If you click in the Window menu, you'll see three sub-items under the "Hello World - (Hello World)" item. The one selected in **Figure 7** is the Build window. Under that is the Project window (in this case, listing the main.c source code). Under that is the Run window showing the output of the program execution.

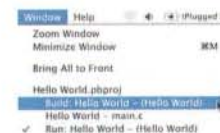


Figure 7. Project Builder's Window menu.

You can close any of these windows at any time, though you'll likely want to keep the project window open so you can make changes to your source code and build and run your app.

TILL NEXT MONTH...

Want to play some more? Good! Try making some changes to the source code. Got an old copy of *Learn C on the Macintosh* lying around? Try typing in some of that source code. And if you are really adventuresome, take the debugger for a spin. Hint: The tooltip for the 2nd icon from the left in the project window's toolbar says "Build and debug active executable".

Next month, we'll go through Project Builder in a bit more detail and go through the debugger as well. It's good to be back – Thanks for reading!

Microsoft



Mac and PCs have never been so compatible.

Microsoft® Office v. X lets Mac users effortlessly open, edit, and save any Office file, to make working with PCs a breeze. Complete with easy-to-use, exclusive Mac tools that simplify complex tasks. And it's built for Mac OS X, so it's the most reliable, easygoing Office yet. www.officeformac.com to download a free 30-day trial of Office v. X today.



Microsoft®
Office:mac
v. X

By Andrew C. Stone

The Ins and Outs of Drag and Drop

The most copied feature of the Mac OS X interface is the ubiquitous drag and drop. When NeXTStep advanced the techniques pioneered at Xerox's Palo Alto Research Center in the late '80's, the way in which people interacted with software was changed forever. The ability to move data and objects seamlessly between windows and applications without any additional steps is the hallmark of a native OS X application. This article will explore some more advanced techniques and some of the issues you might encounter when preparing your interface for drag and drop, as covered in other MacTech articles, such as http://www.stone.com/The_Cocoa_Files/What_a_Drag_.html. We'll cover making an entire window a receptacle for drag and drop, using central control to reduce code, dealing with temporary subviews such as field editors, and auto-swapping Tab views based on the type of data being dropped.

Many of Stone Design's applications fit into the category of "just drag and drop and you're done," such as PStill, GIFfun, PackUpAndGo, DOctor and SliceAndDice. Taking PStill as an example, the user just drags a file onto the PStill window or application tile in the Dock or the Finder to convert the file to or redistill it as PDF:



Drag files onto Dock or Window

The strategy I like to employ is to make the entire window a valid drag target by subclassing `NSWindow` or `NSPanel`, and forwarding the actual methods to the window's delegate:

```
@interface NSObject(implement_this_in_delegate)
- (void)registerTypesForPanel:(NSPanel *)panel;
@end

@interface SDDragInPanel : NSPanel
{}
@end

@implementation SDDragInPanel

- (void)awakeFromNib
{
    [[self delegate] registerTypesForPanel:self];
}

- (unsigned int) draggingEntered:sender
{
    return [[self delegate] draggingEntered:sender];
}

- (unsigned int) draggingUpdated:sender
{
    return [[self delegate] draggingUpdated:sender];
}

- (BOOL) prepareForDragOperation:sender
{
    return [[self delegate] prepareForDragOperation:sender];
}

- (BOOL) performDragOperation:(id <NSDraggingInfo>)sender
{
    return [[self delegate] performDragOperation:sender];
}

@end
```

Typical code for delegate would be:

```
// Dragging stuff:
- (NSArray *)acceptableDragTypes
{
    return [NSArray
        arrayWithObjects:NSFileNamesPboardType,nil];
}

- (void)registerTypesForPanel:(NSPanel *)panel;
{
    [panel registerForDraggedTypes:[self
        acceptableDragTypes]];
}

- (unsigned int)draggingEnteredOrUpdated:(id
```

Andrew Stone is founder, janitor and chief computer scientist at Stone Design, www.stone.com.


```

<NSDraggingInfo>)sender {
    // we want to ignore drags originating in our own window:
    if ([sender draggingSource] == dragWellView) return
    NSDragOperationNone;
    else {
        unsigned int sourceMask = [sender
        draggingSourceOperationMask];
        NSPasteboard *pboard = [sender draggingPasteboard];
        NSString *type = [pboard
        availableTypeFromArray:[self acceptableDragTypes]];
        if (type) return sourceMask;
        return NSDragOperationNone;
    }
}

- (unsigned int)draggingEntered:(id <NSDraggingInfo>)sender
{
    return [self draggingEnteredOrUpdated:sender];
}

- (unsigned int)draggingUpdated:(id <NSDraggingInfo>)sender
{
    return [self draggingEnteredOrUpdated:sender];
}

- (BOOL)performDragOperation:(id <NSDraggingInfo>)sender
{
    NSPasteboard *pboard = [sender draggingPasteboard];
    NSString *type = [pboard availableTypeFromArray:[self
    acceptableDragTypes]];
    BOOL loaded = NO;
    id ts = nil;
    if (type) {
        if ([type isEqualToString:NSFileNamesPboardType]) {
            NSArray *files = [pboard
            propertyListForType:NSFileNamesPboardType];
            unsigned i = [files count];
            while (i-- > 0) {
                NSString *f = [files objectAtIndex:i];
                if ([[self acceptableFileTypes]
                containsObject:[f pathExtension]] && (ts = [SomeObject
                objectWithContentsOfFile:f])!=nil) {
                    loaded = YES;
                    break;
                }
            }
        }
    }
    return loaded;
}

- (BOOL)prepareForDragOperation:sender
{
    return YES;
}

```

Be sure to register the view for the accepted types. A good place to do this is in - (void)awakeFromNib. This method is called on any object instantiated in a NIB (NeXT Interface Builder) file that has an implementation of awakeFromNib after all the objects are created and linked up, but before the window appears on screen. By implementing a single method acceptableDragTypes that returns which types you actually accept, you can avoid out-of-synch code when you add more types to open later:

```

[panel registerForDraggedTypes:[self
acceptableDragTypes]];

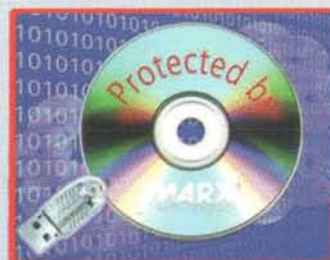
```

So we are done, right? Not quite, because of the way NSTextFields work. When you click or tab into a text field, a shared NSTextView is inserted into the view hierarchy. When the user drags a file over any part of the window that doesn't have an active textfield, the draggingEntered works as planned. But when you pass over the active text field, the NSTextView's drag validation methods come into play. The solution is to subclass NSTextView to also forward the methods to your window's delegate - or just to the window, since the window will forward on to the delegate:



MARX® CRYPTO-BOX USB Security Key For Macintosh

Use the CRYPTO-BOX® USB to secure data, networks, or software from unauthorized use.



Encryption

Access Control

License Control

User Limits



Supports all Macintosh Systems with USB including OS 8.6-9.1, & MAC OS X (CFM, Mach-O, and apps in classic), Windows in MAC OS 9 and Virtual PC 4. Metrowerks CodeWarrior C/C++ and Apple Project Builder C/C++ samples available

Contact

www.marx.com

MARX Software Security

2900 Chamblee-Tucker Road N.E., Bldg. 9
Atlanta, GA 30341, USA

☎ 1-800-MARX-INT

☎ 1-770-986-8887

fax 1-770-986-8891

info@marx.com

MARX Software Security GmbH

Vohburger Strasse 68

D-85104 Wackerstein

Germany

☎ (+49) 8403/9295-0

fax (+49) 8403/1500

contact-de@marx.com



Securing the Digital World™

Time Track



Keep track of every second of your time and bill for it with Time Track! Built in instructions make it easy to use. It is a simple way to manage your billable time for multiple projects and create a web page to show to your clients. Only \$24.95 per single user license per platform. Finally! A versatile time tracking solution for Macintosh, Windows, and Palm!

www.trinfinitysoftware.com



Eudora Internet Mail Server (EIMS) 3.1 is the latest version of the most popular Internet mail server for the Macintosh. If you need to handle email for a dozen users, or thousands of users, EIMS is a reliable and easy to use solution.

EIMS 3.1 is available for US\$400.00, there are no limits on the number of users that can be added, and free email support is included.

For more information, see
<http://www.eudora.co.nz/>



GraphicConverter converts pictures to different formats. Also it contains many useful features for picture manipulation.

See www.lemkesoft.com
for more information.

Suite Deal

A suite of must-have
utilities for Mac OS X
\$170 value for \$49.99

Ten for X delivers fully-registered favorites in one convenient package at one great price. Plus, OS X tips from the pros and dozens of trial versions from top-names like Adobe and Microsoft are included FREE! Visit www.aladdinsys.com/tenforx

Available at **DEPOT**
www.devdepot.com



Ten for X

- Alarm Clock S.E.
- ExecutiveSync
- FruitMenu
- iClean
- ideaSpiral X
- LaunchBar
- Limewire Pro
- piPop
- Pseudo
- PrintMagic X
- WindowShade X
- Sounds



Aladdin products are available from your favorite Mac dealer or catalog and at shop.aladdinsys.com
www.aladdinsys.com • www.Stuffit.com • www.TenforX.com

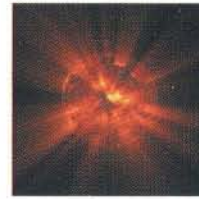
Copyright ©2002, Aladdin Systems, Inc. All rights reserved. Ten for X is a trademarks of Aladdin Systems, Inc. The Aladdin Logo is a registered trademark of Aladdin Systems, Inc. Other brand and product names are the trademarks or registered trademarks of their respective owners.



Watson 1.5

Winner of the 2002 Apple Design Award for Most Innovative Mac OS X Product! An Aqua experience for the most useful Web services: TV & Movie Listings, Reference, Translation, Stocks, Flights, Package Tracking, and more. With an open architecture for third-party tools as well. Download the demo now!

www.karelia.com/watson



Trapcode - plug-ins for Adobe® After Effects®

Trapcode Shine is a fast light effect plug-in. The effect looks very much like volumetric light, but is actually a 2D effect. There are special controls to make shimmering lights and numerous coloring modes. This is an effect that you see everyday on TV and in many movie titles. Shine is available for Mac, Mac OSX and Windows.

www.trapcode.com



By TLA Systems

The Dock with more than one dimension.

Create multiple docks of any size, assign hot keys and even put the Trash back on the Desktop. A flexible and feature-laden tool for power-users. Runs natively on Mac OS X and 9 in five languages.

"...you made the switch to OS X a lot easier for me..." - Bob LeVitus

"...DragThing can rightfully be called an indispensable aid to working with your Mac..." - MacUser UK

Download a copy now from www.dragthing.com.

The Journal of Macintosh Technology & Development

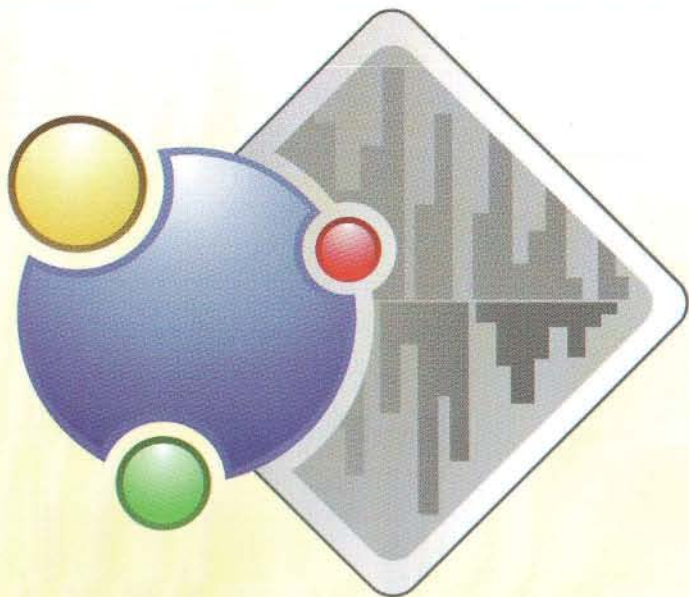
MacTech

Got a great product sold through Kagi?

Promote your product through the very cost effective Kagi Showcase in MacTech Magazine.

For more information, contact us at

adsales@mactech.com



IPNetMonitorX

- The Swiss Army Knife of Mac OS X Internet Diagnostic Tools
- 15 Fully Integrated Tools—Monitor, Link Rate, Address Scan, TCP Dump and more
- Troubleshoot and Solve Network Problems
- Responsive, Intuitive, Easy to Use Interface
- Fast Multi-Threaded Architecture—see network behavior as it happens
- 21-Day Fully Functional Free Trial



Download Your Copy Now at

www.sustworks.com/mac

SUSTAINABLE

Softw**orks**

Tools for Internet Travel



```
#import <Cocoa/Cocoa.h>

@interface DragTextView : NSTextView
{
}
@end
#import "DragTextView.h"

@implementation DragTextView
// override drag stuff...
- (id)initWithFrame:(NSRect)r
{
    [super initWithFrame:r];
    [self registerForDraggedTypes:[[[self window] delegate]
acceptableDragTypes]];
    // this is so TAB and RETURN end editing
    // instead of being inserted into the field:
    [self setFieldEditor:YES];
    return self;
}

// note we just pass it up to the window:

- (unsigned int) draggingEntered:sender
{
    return [[self window] draggingEntered:sender];
}

... etc. just passing on the method to the window

@end
```

Now we have our custom text view, but how do we make sure our text view is used in place of the standard text view? We can't set z in Interface Builder, but we can code it. If a window's delegate implements a method called `-windowWillReturnFieldEditor:(NSWindow *)sender toObject:(id)client`, the Appkit code will call this method and use the text view it returns if non-nil, otherwise it uses a standard text view set in field editor mode.)

```
// add textView as an iVar to the NSWindowController subclass which controls the window
- (id)windowWillReturnFieldEditor:(NSWindow *)sender
toObject:(id)client {
    if (sender == [self window]) {
        if (![textView] textView = [[DragTextView
alloc] initWithFrame:[myField bounds]]);
        return textView;
    }
    return nil;
}
```

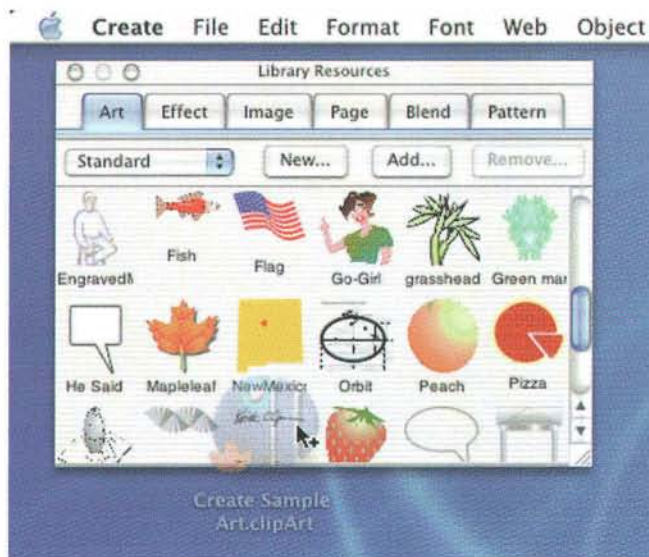
At this point, our interface is ready to accept the correct files and data at any location in the window.

There's one final issue: what if a user can drag a file out of your interface (for example, in PStill, you can drag the distilled PDF file out of the "drag out well") and that file type can also be dragged in to the application (for example, PStill accepts PDF files as input). A user might start a drag out of the application, change her mind, and drop the file back onto the application window. In this case, the application should probably not process the file. Therefore, the window delegate should check the `draggingSource` to make sure it's not a component of the window itself. This is why we have this line in the `draggingEnteredOrUpdated` code above:

```
if ([sender draggingSource] == dragWellView) return
NSDragOperationNone;
```


AUTOSWAPPING TAB VIEWS

The concept of filtering the dragging methods through the window's delegate can be very useful when your window contains an NSTabView with different acceptable types in each view. In Create®, for example, there is a resources library which can accept art, images, effects, blends, patterns and pages:



Create® lets you store many different types of resources - and it will swap to the correct tab view as necessary

Each of these tabviews has an NSScrollView, which contains an NSMatrix. When a user drags in a certain type that is not correct for the current view, but is acceptable in another one of the tab views, the tab view should automatically switch to the other view so that the drag can drop successfully in the right place. We do this by first checking if we can deal with it - and if not, we'll ask the window controller (which keeps track of the other views) to check the other resource managers. Note we also don't want to accept drags that start from this particular resource's matrix:

```
- (unsigned int)draggingEntered:(id <NSDraggingInfo>)sender
{
    return [self draggingEnteredOrUpdated:sender
        checkOthers:YES];
}

- (unsigned int)draggingUpdated:(id <NSDraggingInfo>)sender
{
    return [self draggingEnteredOrUpdated:sender
        checkOthers:YES];
}

- (unsigned int)draggingEnteredOrUpdated:(id
<NSDraggingInfo>)sender checkOthers:(BOOL)checkOthers
{
    if ([sender draggingSource] == dragMatrix) return
    NSDragOperationNone;
    else {
        NSPasteboard *pboard = [sender draggingPasteboard];
        NSString *type = [pboard
        availableTypeFromArray:[self acceptableDraggedTypes]];
```

PRIMEBASE 4.0

New Launcher for OS X

Installation has never been easier on Mac OS X with the PrimeBase Launcher.

Download the Launcher now
and get started with PrimeBase

www.primebase.com

**developer keys
available
free of charge**

**PRIMEBASE DATABASE SERVER AND
PRIMEBASE APPLICATION SERVER
AVAILABLE ON THE MOST POPULAR PLATFORMS**

- Completely cross-platform
- Full-text searching and indexing
- Mac OS classic & X
- Mac OS X Server
- Linux
- Solaris
- IBM AIX
- all Windows platforms

**Develop on a Mac and deploy without any
additional effort on any platform you want.**

 **PRIMEBASE**

SNAP Innovation GmbH
Altonaer Poststraße 9a
D-22767 Hamburg / Germany
www.primebase.com
e-mail: info@primebase.com
Fon: ++49 (40) 389 044-0
Fax: ++49 (40) 389 044-44


```

        if (type) {
            unsigned int sourceMask = [sender
draggingSourceOperationMask];

            if ([type
isEqualToString:NSFileNamesPboardType]) {
                NSArray *filenames = [pboard
propertyListForType:NSFileNamesPboardType];
                if ([filenames count] == 1) {
                    NSString *filename = [filenames
objectAtIndex:0];
                    if ([[self resourceClass] fileTypes]
containsObject:
                        [filename pathExtension])
                        return sourceMask;
                    } else return sourceMask;
                }
            }
            if (checkOthers) return [_controller
draggingEnteredOrUpdated:sender];
            else return NSDragOperationNone;
        }
    }
}

```

The `_controller`'s implementation might look something like this:

```

- (unsigned int)draggingEnteredOrUpdated:(id
<NSDraggingInfo>)sender
{
    int i, c = [_resourceSources count];
    unsigned int returnValue;
    for (i = 0; i < c; i++) {
        ResourceSource *res = [_resourceSources
objectAtIndex:i];
        if (res == _currentSource) continue; // already checked!
        if ((returnValue = [res
draggingEnteredOrUpdated:sender checkOthers:NO]) !=
NSDragOperationNone) {
            [self showResourceSourceNamed:[res
resourceSourceName]];
            return returnValue;
        }
    }
    return NSDragOperationNone;
}

- (void)showResourceSourceNamed:(NSString *)name
{
    [tabView selectTabViewItemAtIndex:[tabView
indexOfTabViewItemWithIdentifier:name]];
}

```

Because the matrix may not fill the scroll view entirely, we'll also have to subclass the scroll view to forward `draggingEntered` methods to the matrix. To the end user, the entire scroll view is seen as the target, not just the matrix!

```

- (unsigned int)draggingEntered:(id <NSDraggingInfo>)sender
{
    return [[self documentView]draggingEntered:sender];
}

```

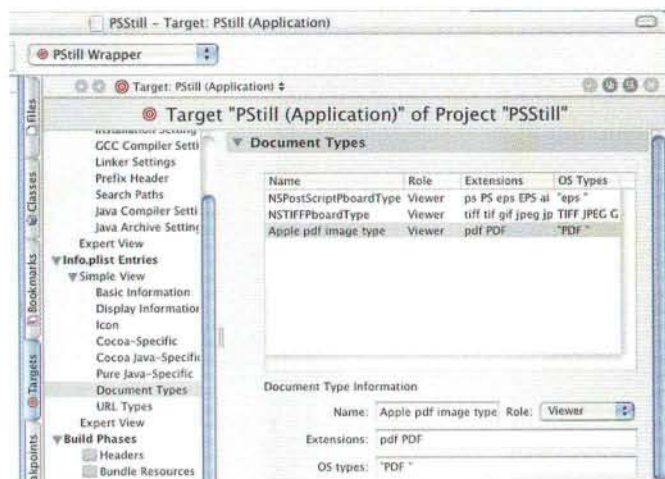
... etc. for all the other methods.

APPLICATION TILE DRAG SUPPORT

You only have to perform a few tasks to add support for drag and drop to your Application icon and its Dock tile. First, you'll need to alert the system of the valid file types handled by your application. Then, you'll implement a method in the Application's delegate subclass which calls the actual method to

deal with that file type.

First, add information about which files can be opened by your application in Project Builder's application Target Inspector, in the "Document Types" pane:



Be sure to add the file types that your application can open in Project Builder

Second, set your application's delegate. You can do this programmatically with `NSApplication`'s `setDelegate:`. Or, you can use Interface Builder: (a) instantiate an object of your delegate class in your main NIB file, and (b) connect the File's Owner instance variable "delegate" to this new object.

Third, implement a single method in your delegate's class:

```

- (BOOL)application:(NSApplication *)sender
openFile:(NSString *)path
{
    MyDocument *doc = [[NSDocumentController
sharedDocumentController] openDocumentWithContentsOfFile:path
display:YES];
    return doc;
}

```

Now, not only will the dock tile accept drag and drop, but Finder will display your application as a choice for opening that kind of document.

CONCLUSION

If you want your application to really sing, be sure users can take full advantage of drag and drop everywhere! Not only will it make program interaction easier and more fun, it makes demoing the app more spectacular!



New OS. New Software. New Installer.

Introducing InstallAnywhere - Mac OS X Edition, the new power tool for your Mac OS X software installation. Stop using yesterday's tired old installer tools. Now, you can create industrial strength installers that are flexible, intuitive, and royalty-free. Your software will look better than ever and install perfectly, every time.

InstallAnywhere supports all Mac OS X features, like user authentication, file permissions, installing icons into the Dock, and offers a fully customizable Aqua look and feel.

Software innovators like Adobe, Apple, Borland, Gracion, LimeWire, and Sun already depend on Zero G for their software installation needs. See for yourself why InstallAnywhere - Mac OS X Edition is the new power tool for your software.

InstallAnywhere - The Industrial Strength Installer From Zero G.

Download a free trial version from <http://www.ZeroG.com/goto/mac>



DISCOVERY AHEAD

EXPLORE, EXPERIENCE, DECIDE.

Your Future is at COMDEX.

COMDEX Fall 2002 is your best opportunity to experience, explore, and learn about the latest product innovations, leading vendors, and powerful new business strategies.

- FREE access to the exhibit floor
- FREE entrance to the Great Debates and Hot Spots
- FREE admission to the new Digital Lifestyles Conference and SuperSession
- Content-rich Educational Programs for business and technical professionals

Register today!

**Visit www.comdex.com/fall to register or call 888-568-7510.
Use Priority Code HCMG and Coupon Code 342.**



PRESENTS

COMDEX®
FALL 2002

THE GLOBAL TECHNOLOGY MARKETPLACE

OFFICIAL CORPORATE SPONSORS OF KEY3MEDIA GROUP



Official Card of COMDEX



Official Automotive Sponsor



New York Stock Exchange

Copyright © 2002 Key3Media Events, Inc., 5700 Wilshire Boulevard, Suite 325, Los Angeles, CA 90036-3659. All Rights Reserved. CO02-12171 10/02 Key3Media, COMDEX, and associated design marks and logos are trademarks owned or used under license by Key3Media Events, Inc., and may be registered in the United States and other countries. Other names mentioned may be trademarks of their respective owners.

Conference Overview

Make the most of your COMDEX experience with our affordable Educational Programs.

COMDEX Educational Programs keep you at the forefront of technology and are designed to provide the specific information and strategies you need to maximize the return on your technology investments.

Business Technology Conference

Acquire in-depth information on key technologies, essential business strategies, and action-oriented solutions to pressing IT issues such as ROI, security, Web services, wireless, storage, business continuity, and project management.

TechCentric Conference

Get innovative solutions for today's critical development challenges, explore new tools, and learn about the latest network and infrastructure technologies.

Real-Time Enterprise Conference

Discover how ERP, CRM, SFA, procurement, logistics, and supply chain databases can come together to create new ways to link critical business systems, deploy technology, and empower the workforce.

The COMDEX IT Executive Symposium,

presented in partnership with **BusinessWeek**

Hear from visionary technologists, innovative companies, and executive practitioners who are reconciling the euphoria of the past with the real-world pragmatism of the present to make breakthrough technologies pay off.

FORTUNE Small Business Forum

This survival guide will enable small businesses to use the Web to crack new markets, generate sales, and develop winning marketing strategies.

All Conference registrants receive FREE access to the Exhibits, VIP seating at Keynotes, Great Debates, ZDNet Unplugged Interview, and the Digital Lifestyles Conference, plus a COMDEX conference bag.

Be sure to check out our complete Conference, Tutorial, and Certification schedule at www.comdex.com/fall.

Keynotes—Free to all COMDEX attendees



BILL GATES
Microsoft Corporation



PETER CHERNIN
News Corporation and
Fox Group



CARLY FIORINA
Hewlett-Packard
Company



BRIAN HALLA
National Semiconductor
Corporation



SCOTT McNEALY
Sun Microsystems, Inc.



CARLOS BONILLA
National Economic
Council



HECTOR DE J. RUIZ
AMD



STEPHEN WOLFRAM
Wolfram Research

**Gain full access
with a Flex Pass.**

Enjoy the flexibility of attending any of the COMDEX Educational Programs—including all of the COMDEX Conferences and Tutorials. (Does not include Microsoft and Cisco Professional IT Certifications.)

2002

COMDEX EDUCATIONAL PROGRAMS: November 16–21, 2002
LAS VEGAS CONVENTION CENTER | LAS VEGAS, NEVADA

COMDEX EXHIBITION: November 18–22, 2002



By Dan Wood, Alameda CA

Table Techniques Taught Tastefully (part 3)

Using NSTableView for Real-World Applications

INTRODUCTION

This is the last of a series of articles about the wonderful NSTableView class in Cocoa. While the first part went over the basics, and part the second got your hands dirty, this last part is where we pull out all the stops and do some really cool things with tables that will make you the envy of all the Cocoa programmers on your block.

In this article, we'll show you how to give your tables those trendy blue and white alternating stripes and vertical grid-lines that you see on programs like iTunes. We'll make a subclass of NSTableView that merges certain cells together across multiple columns, suitable for display of a time schedule. We'll make a custom cell to indicate relevance, like you see when you search in Apple's Mail program or the Finder. And finally, we'll see how to animate the sorting of a table, like you see in iChat. (Warning: There will be math in the last segment!)

Be sure to follow along with the "TableTester" application (downloadable at www.karelia.com/tabletester/), a program showing off most of the table features described in this series. It contains the source code corresponding to the techniques in this article as well as those in the first two parts, in case you missed them the first time around.

STRIPED TABLE ROWS

If you want your table display to look like one of Apple's applications like iTunes, or just to make your list more readable, you may want to consider alternating row background colors. At first glance, it seems that the best way to accomplish this is to intercept the `tableView: willDisplayCell: forTableColumn: row: delegate` message and set the cell's background color depending on whether the row is even or odd. Unfortunately, this only stripes the cells with data, rather

than the entire table; it also works only for text cells, not button or image cells.

A better approach is to create a subclass of NSTableView and override `highlightSelectionInClipRect:` (**Listing 1**) to draw the stripes. This method draws stripes in the background by alternating between the "even" color of light blue and the "odd" color of white.

| Saw | Sign | Name |
|-------------------------------------|---|----------|
| <input type="checkbox"/> |  | Julie |
| <input checked="" type="checkbox"/> |  | Sandra |
| <input type="checkbox"/> |  | Reginald |
| <input type="checkbox"/> |  | Hakim |
| <input checked="" type="checkbox"/> |  | Rich |
| <input type="checkbox"/> |  | Sophia |
| <input type="checkbox"/> |  | George |
| <input type="checkbox"/> |  | Darnell |
| <input type="checkbox"/> |  | Roxanne |
| <input checked="" type="checkbox"/> |  | Korvax |

Figure 1. Alternating Rows and Vertical Grids

Listing 1: StripedTableView.m

```

highlightSelectionInClipRect:
Display the background for the table in the given clipping rectangle.

- (void)highlightSelectionInClipRect:(NSRect)clipRect
{
    NSColor *evenColor // empirically determined color, matches iTunes etc.
    = [NSColor colorWithCalibratedRed:0.929

```

Dan Wood, the son of an organic cropduster pilot and a neurosurgeon, grew up in Amish Pennsylvania, back in the roaring twenties. As a child, he watched *Lost in Space* and *Powerpuff Girls* on TV. After graduating with a degree in Chocolatology from the University of Hershey, Dan joined the Peace Corps, teaching American Sign Language to underprivileged dolphins. He is currently a road crew foreman for the California Department of Transportation (CalTrans), and has written a successful application in Cocoa called *Watson*. Dan thanks Chuck Pisula at Apple for his technical help with this series, and acknowledges online code fragments from John C. Randolph, Stéphane Sudre, Ondra Cada, Vince DeMarco, Harry Emmanuel, and others. You can reach him at dwood@karelia.com.



Save 20%
off these titles at

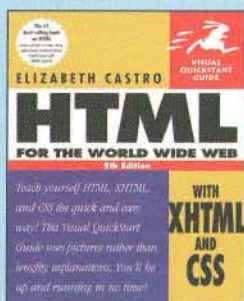
BORDERS®

*Sale begins
November 3, 2002

Visual QuickStart Guides

Get up and running quickly!

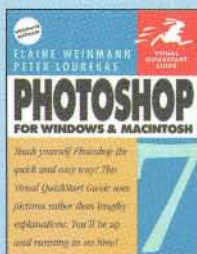
Our *HTML with XHTML and CSS: Visual QuickStart Guide* is better than ever...now in its 5th Edition!



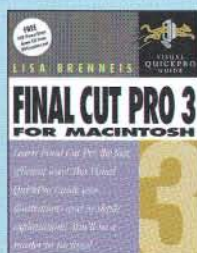
HTML for the World Wide Web, Fifth Edition, with XHTML and CSS: Visual QuickStart Guide
Elizabeth Castro
0-321-13007-3 • \$21.99

The Web is changing how it does business and so should you! If you're still coding like it was 1999, you need to update your HTML skills with Elizabeth Castro's *HTML for the World Wide Web, Fifth Edition, with XHTML and CSS: Visual QuickStart Guide*. This latest edition of the original book on HTML will have you creating complex, dynamic sites that look good across all browsers and platforms—including handheld devices and cell phones—in no time!

The fastest way to learn!



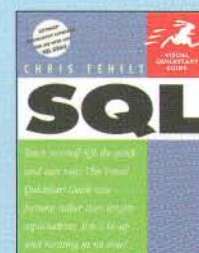
Photoshop 7 for Windows and Macintosh: Visual QuickStart Guide
Elaine Weinmann and Peter Lourekas
0-201-88284-1 • \$24.99



Final Cut Pro 3 for Macintosh: Visual QuickPro Guide
Lisa Brenneis
0-321-11583-X • \$29.99



Macromedia Dreamweaver MX for Windows and Macintosh: Visual QuickStart Guide
J. Tarin Towers
0-201-84445-1 • \$24.99



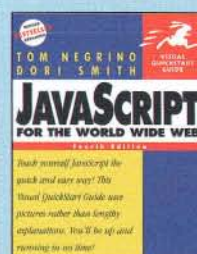
SQL: Visual QuickStart Guide
Chris Fehily
0-321-11803-0 • \$21.99



Mac OS X 10.2: Visual QuickStart Guide
Maria Langer
0-321-15801-6 • \$21.99



Macromedia Flash MX for Windows and Macintosh: Visual QuickStart Guide
Katherine Ulrich
0-201-79481-0 • \$24.99



JavaScript for the World Wide Web, 4th Edition: Visual QuickStart Guide
Tom Negrino and Dori Smith
0-201-73517-2 • \$19.99



Macromedia Flash MX Advanced for Windows and Macintosh: Visual QuickPro Guide
Russell Chun
0-201-75846-6 • \$29.99


```

        green:0.953 blue:0.996 alpha:1.0];
        NSColor *oddColor = [NSColor whiteColor];

        float rowHeight
        = [self rowHeight] + [self intercellSpacing].height;
        NSRect visibleRect = [self visibleRect];
        NSRect highlightRect;

        highlightRect.origin = NSMakePoint(
            NSMinX(visibleRect),
            (int)(NSMinY(clipRect)/rowHeight)*rowHeight);
        highlightRect.size = NSMakeSize(
            NSWidth(visibleRect),
            rowHeight - [self intercellSpacing].height);

        while (NSMinY(highlightRect) < NSMaxY(clipRect))
        {
            NSRect clippedHighlightRect
            = NSIntersectionRect(highlightRect, clipRect);
            int row = (int)
            ((NSMinY(highlightRect)+rowHeight/2.0)/rowHeight);
            NSColor *rowColor
            = (0 == row % 2) ? evenColor : oddColor;
            [rowColor set];
            NSRectFill(clippedHighlightRect);
            highlightRect.origin.y += rowHeight;
        }

        [super highlightSelectionInClipRect: clipRect];
    }
}

```

To mimic the iTunes look even further, you may want to draw a grid, but only the vertical lines between columns, not the horizontal lines between rows. So we override `drawGridInClipRect:` and provide our own implementation (**Listing 2**) that draws light gray vertical lines.

Listing 2: StripedTableView.m

```

drawGridInClipRect:
Draw the grid, but only the vertical lines.

- (void)drawGridInClipRect:(NSRect)rect
{
    NSRange columnRange = [self columnsInRect:rect];
    int i;
    [[NSColor lightGrayColor] set];

    for ( i = columnRange.location ;
          i < NSMaxRange(columnRange) ;
          i++ )
    {
        NSRect colRect = [self rectOfColumn:i];
        int rightEdge
        = (int) 0.5 + colRect.origin.x + colRect.size.width;
        [NSBezierPath strokeLineFromPoint:
            NSMakePoint(-0.5+rightEdge, -0.5+rect.origin.y)
            toPoint:
            NSMakePoint(-0.5+rightEdge, -0.5+rect.origin.y +
            rect.size.height)];
    }
}

```

Voila! Striped tables, as in Figure 1. Now you can write the next iApp!

MERGING TABLE CELLS TOGETHER

So far, all of the subclassing of `NSTableView` that we've done in this series are pretty straightforward, and modify the default behavior only subtly. But what happens when the class is radically subclassed? Well, one example of this is Cocoa's own `NSOutlineView`, a subclass that barely resembles its parent in the way that it structures and presents its contents. In this

segment, we'll try something a bit less ambitious, but significant nevertheless.

The challenge is this: to have a table view in which certain columns are merged together with their neighboring columns. An example application would be a daily schedule in which appointments take a variable amount of time. You want the cells to span across multiple columns, not constrained to individual columns. (Readers familiar with HTML can equate this to the "colspan" attribute of a `<TD>` tag.)

There are two sides to making this work. One is to modify the controller code that provides the data for the table to display; the other is to implement the view (the `NSTableView` subclass, which we call `MergedColumnTableView`) to display the data provided by the controller.

For the controller, we invent a new method for an informal protocol for your controller to define:

```

(int)tableView:(NSTableView *)tableView
    spanForTableColumn:(NSTableColumn *)tableColumn
    row:(int)row;

```

Our implementation should return 1 if the cell is one column wide (the usual case); 0 if no data is to be shown in the column (generally the case if it is to the right of a multiple-cell-spanning table), and a number greater than 1 if the cell is to span more than one column to the right. Because we pass in a pointer to the table view, this method can be used even if there is more than one table that your controller controls; because we pass in a row number, each row can have differ in its presentation.

The `TableTester` application accompanying this article reads in a sample "class schedule" from a property list file, and implements the standard `NSTableView` data source methods of `numberOfRowsInTableView:` and `tableView:objectValueForTableColumn:row:` as well as the spanning method for this protocol. We won't be examining the controller implementation in-depth here, since its very dependent on the data structure. Unlike a typical table display, where an array of dictionaries will usually suffice, the data to display is more complex. So if you need to display data that spans multiple columns, you can take advantage of `MergedColumnTableView`, but you are on your own for implementing the controller. (This is yet another reason why it's good to partition the view, the controller, and the model!)

How the `MergedColumnTableView` subclassing works, on the other hand, is significant, because it may help reveal techniques that you can use for other table subclassing needs. We override three methods of `NSTableView`: `frameOfCellAtColumn:row:`, to override the rectangle for a given cell to take the column spanning into account; `drawRow:clipRect:`, to manage the drawing of all the columns in a row; and `drawGridInClipRect:`, to draw the table grid lines in such a way that merged cells have no grid line between them. (Each of those methods provides a special behavior if the data source implements our additional method; if not, the superclass provides the default behavior.)

The most important override is `frameOfCellAtColumn:row:`. (See **Listing 3**.) This is the method that calculates the rectangle

associated with a given cell for each column and row. Since our goal is to make the cells wider than they would normally be, this is the logical place to modify the default table behavior. All we have to do is find out the number of columns that the given cell should span, and return an appropriate rectangle. If the column span is zero, we return an empty rectangle, NSZeroRect. If the column span is one, we let the superclass calculate the rectangle, since nothing is different about a cell with a column span of one. If the column span is greater than one, we collect up all the rectangles of the current cell and the cells to the right by invoking the superclass's method, merging the rectangles together into one bigger rectangle.

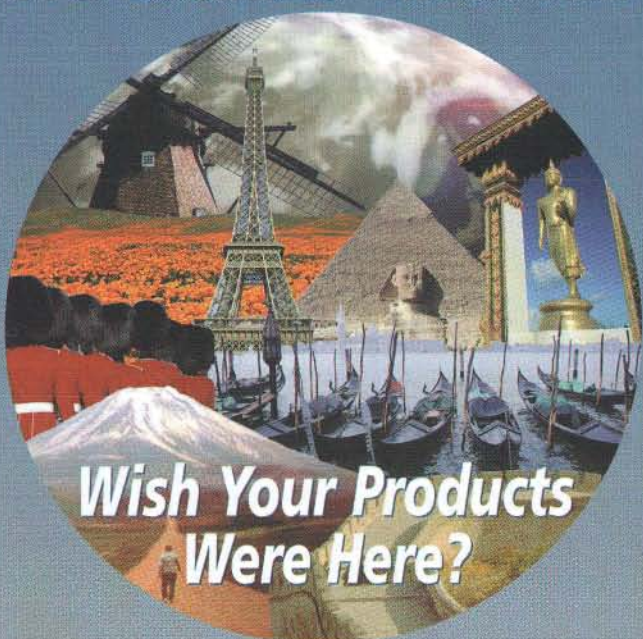
Listing 3: MergedColumnTableView.m

frameOfCellAtColumn:row:
Return the rectangle for the given cell. This may behave like its superclass, or it may return an empty rectangle or a wider rectangle if the table's column span is not one.

```
- (NSRect)frameOfCellAtColumn:(int)column row:(int)row
{
    int colspan;
    if (![[self dataSource]
        respondsToSelector:
            @selector(tableView:spanForTableColumn:row:)])
    {
        return [super frameOfCellAtColumn:column row:row];
    }
    colspan = [[self dataSource]
        tableView:self
        spanForTableColumn:
            [[self tableColumns] objectAtIndex:column]
        row:row];
    if (0 == colspan)
    {
        return NSZeroRect;
    }
    if (1 == colspan)
    {
        return [super frameOfCellAtColumn:column row:row];
    }
    else // 2 or more, it's responsibility of data source to provide reasonable number
    {
        NSRect merged
            = [super frameOfCellAtColumn:column row:row];
        // start out with this one
        int i;
        for (i = 1; i < colspan; i++) // start from next one
        {
            NSRect next
                = [super frameOfCellAtColumn:column+i row:row];
            merged = NSUnionRect(merged,next);
        }
        return merged;
    }
}
```

The above override covers most of the needed functionality, but a couple of subtle items remain. When horizontal scrollbars are used in the table, you will find that some table cells don't get drawn on the left edge of the table. This is because the standard NSTableView draws only the cells that are currently visible within the NSScrollView, and it doesn't realize that a cell spanning multiple columns needs to be drawn even if it starts to the left of the visible rectangle. (See **Figure 2**.) So we override **drawRow: clipRect:** (**Listing 4**) to look at the leftmost column, and if the cell there has a column span of zero, it needs to "back up" to the left and find the cell that spans multiple columns. Once it finds that cell, it expands the clipping rectangle so that the wide cell will be drawn and thus appear in the visible region.

Classic Or Cocoa Applications? We Localize Them All!



- Translation • Engineering
- Localization • Project
- Desktop Publishing • Management
- Quality Assurance

To receive your FREE copy of
**The Guide to Translation and
Localization - Preparing Products
for the Global Marketplace:**

Call: **1-800-878-8523**

Fax: **1-503-419-4873**

Email: **info@lingosys.com**

Or visit: **www.lingosys.com**



15115 SW Sequoia Pkwy. #200 Portland, OR 97224

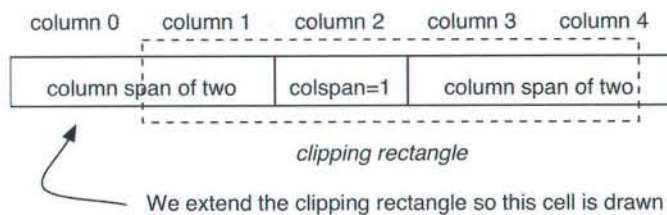


Figure 2. Missing Columns

Listing 4: MergedColumnTableView.m

```

typeAheadString:inTableView:
Actually select the appropriate row based upon the string that has been typed.

- (void)drawRow:(int)inRow clipRect:(NSRect)inClipRect
{
    NSRect newClipRect = inClipRect;
    if ([[self dataSource]
        respondsToSelector:
            @selector(tableView:spanForTableColumn:row:)])
    {
        int colspan = 0;
        int firstCol
            = [self columnsInRect:inClipRect].location;
        // Does the FIRST one of these have a zero-colspan? If so, extend range.
        while (0 == colspan)
        {
            colspan = [[self dataSource]
                tableView:self
                spanForTableColumn:[self tableColumns]
                objectAtIndex:firstCol
                row:inRow];
            if (0 == colspan)
            {
                firstCol--;
                newClipRect = NSUnionRect(newClipRect,
                    [self frameOfCellAtColumn:firstCol row:inRow]);
            }
        }
        [super drawRow:inRow clipRect:newClipRect];
    }
}

```

| Day | 8:00 | 9:00 | 10:00 | 11:00 | 12:00 | 13:00 | 14:00 | 15:00 |
|-----------|----------------------------------|------|-------------------------------------|-------|---------------------------|-----------------------------------|-------------------------------|----------------------|
| Monday | Cocoa Programming 200 Hilegas | | Databases Miller | | Linguistics 180 Lakoff | | | Physics Lab Hayne |
| Tuesday | Cocoa Programming 200 Hilegas | | Environmental Science 101 Carson | | | | | |
| Wednesday | Cocoa Programming 200 Hilegas | | Databases Miller | | Linguistics 180 Lakoff | Screenwriting 58 Straczynski | | |
| Thursday | Cocoa Programming 200 Hilegas | | Environmental Science 101 Carson | | | Applied Cryptography Lab Gigli | | |
| Friday | Cocoa Programming 200 Hilegas | | Databases Miller | | Linguistics 180 Lakoff | | Filmmaking Seminar Shabazz | |

Figure 3: The grid lines don't look right.

With the above override, all of the cells will display, but what if you want to display a grid to make the cell sizes clearer? If we use the standard grid, the table would look like **Figure 3**, with vertical lines cutting across our wide cells. So we implement a new grid that takes the column spans into account, drawing vertical lines only before or after cells. If you have a "sparse" display (such as the example class schedule here), it looks even better if you give the table a distinguishing background color. The final result is in **Figure 4**.

| Day | 8:00 | 9:00 | 10:00 | 11:00 | 12:00 | 13:00 | 14:00 | 15:00 |
|-----------|----------------------------------|------|-------------------------------------|-------|---------------------------|-----------------------------------|-------------------------------|----------------------|
| Monday | Cocoa Programming 200 Hilegas | | Databases Miller | | Linguistics 180 Lakoff | | | Physics Lab Hayne |
| Tuesday | Cocoa Programming 200 Hilegas | | Environmental Science 101 Carson | | | | | |
| Wednesday | Cocoa Programming 200 Hilegas | | Databases Miller | | Linguistics 180 Lakoff | Screenwriting 58 Straczynski | | |
| Thursday | Cocoa Programming 200 Hilegas | | Environmental Science 101 Carson | | | Applied Cryptography Lab Gigli | | |
| Friday | Cocoa Programming 200 Hilegas | | Databases Miller | | Linguistics 180 Lakoff | | Filmmaking Seminar Shabazz | |

Figure 4: Grid lines match the column span.

The `drawGridInClipRect:` override (**Listing 5**) determines all of the rows and columns that will need to be drawn for the given clipping rectangle. It then loops first through each row, drawing the horizontal lines using `NSBezierPath` methods, then loops through each row's columns to draw the vertical lines. Similarly to the `frameOfCellAtColumn:row:` override, it builds up the rectangle for a cell spanning multiple columns by growing a rectangle using the `NSUnionRect` operation.

Listing 5: MergedColumnTableView.m

```

typeAheadString:inTableView:
Actually select the appropriate row based upon the string that has been typed.

- (void)drawGridInClipRect:(NSRect)rect
{
    if ([[self dataSource] respondsToSelector:
        @selector(tableView:spanForTableColumn:row:)])
    {
        [super drawGridInClipRect:rect];
    }
    else
    {
        NSRange rowRange = [self rowsInRect:rect];
        NSRange columnRange = [self columnsInRect:rect];
        int row;

        // Adjust column range, always go from zero, so we can gather columns even to
        // the left of what we are supposed to draw.
        columnRange = NSMakeRange(0, NSMakeRange(columnRange));
        [[NSColor grayColor] set];

        for ( row = rowRange.location ;
            row < NSMakeRange(rowRange) ;
            row++ )
        {
            int col = columnRange.location;
            int oldLeftEdge
                = 0.5 + [self rectOfColumn:col].origin.x;
            NSRect rowRect = [self rectOfRow:row];
            // here, frame not the top and not the left, but the bottom
            [NSBezierPath strokeLineFromPoint:
                NSMakePoint(rowRect.origin.x,
                    -0.5+rowRect.origin.y+rowRect.size.height)
                toPoint:
                NSMakePoint(rowRect.origin.x + rowRect.size.width,
                    -0.5+rowRect.origin.y+rowRect.size.height)];

            while ( col < NSMakeRange(columnRange) )
            {
                int colspan = [[self dataSource] tableView:self
                    spanForTableColumn:[self tableColumns]
                    objectAtIndex:col] row:row];
                NSRect gridRect = NSZeroRect;
                if (0 == colspan)
                {
                    col++; // no grid here, move along
                }
                else // Now gather up the next <colspan> rectangles
                {
                    int i, rightEdge, leftEdge;
                    for ( i = 0 ; i < colspan ; i++ )
                    {

```




REAL Software and MacTech present the REALbasic Showcase to highlight some of the fantastic solutions created by REALbasic users worldwide. The showcase illustrates the wide range of applications that developers using REALbasic can create. Some benefit any Mac user, and others are more specific. All of them are seriously cool!

REALbasic is the powerful, easy-to-use tool for creating your own software for Macintosh, Mac OS X, and Windows. It runs natively on Mac OS X as well as earlier versions of the Mac OS. For more information, please visit: <www.realbasic.com>.

The Made with REALbasic program is a cooperative effort between REALbasic users and REAL Software, Inc. to promote the products created using REALbasic and the people who create them. For more information about the Made with REALbasic program, please visit: <www.realbasic.com/realbasic/mwrb/Partners/MwRbProgram.html>.

Extend REALbasic with Advanced Plugins

Spreadsheet Controls:

We provide a wide range of spreadsheet controls for REALbasic, including multistyled and custom rendering spreadsheet controls.

| A | B | C |
|------------------------------|-----------|-----------|
| This is some incredible list | | |
| 1 | Some text | More text |
| 2 | Some text | More text |
| 3 | Some text | More text |
| 4 | Some text | More text |
| 5 | Some text | More text |
| 6 | Some text | More text |
| 7 | Some text | More text |
| 8 | Some text | More text |

- More than 32k of rows.
- Classic, OS X and Win32.
- Accelerated for maximum speed.
- Images in cells.



Cryptography:

We provide data encryption, encoding, compression and hashing for REALbasic.

Supported Algorithms:

- Encryption:
 - e-CryptIt
 - BlowFish (448 bit)
 - AES (Rijndael) (256 bit)
- Encoding:
 - e-CryptIt Flexible
 - Base 64
 - BinHex
 - MacBinary III
 - AppleSingle / Double
 - UUcoding
- Compression:
 - Zip on Strings and streams (.gz)
- Hashing and Checksums:
 - CRC32 / Adler32
 - MD5 / HMAC_MD5
 - SHA / SHA1 / HMAC_SHA1

Other Plugins:

We have many other plugins for REALbasic, including plugins to do advanced MacOS Toolbox tasks and more custom Controls.



Speed up development and make more advanced applications by using plugins ! Get free demos at www.einhugur.com



EinHugur Software
sales@einhugur.com
www.einhugur.com



piPop

Pop-up Hierarchical
File Navigation and Launcher



TelnetLauncher

Bookmark and Launch your
Telnet and SSH sessions



SimpleKeys

Set your Function Keys
to type stuff for you!

piDog Software
<http://www.pidog.com/>

Whistle Blower

Enterprise server monitor and restart utility
whistleblower.sentman.com

Connect to and validate the response from web servers, cgi scripts and over 23 other types of servers.

Send email, pages and perform unattended restarts via MasterSwitch or PowerKey.

Shifts make sure that the person on call when the server goes down is the one who gets the page.

68k, PPC and Carbon


Web based administration lets you check on and restart your servers from anywhere.

Customize your response to an outage with Apple Script.

email us at whistleblower@sentman.com

LIGHTHOUSE

Get Organized with this Time-Saving Internet Assistant



- Search the Web with Free Plugins
- Plugins Manager/Editor
- Local Weather for 160+ Cities
- Calendar Saves Your Important Dates & Events
- World Clock Displays Local Time for 65 Cities
- Note Keeper Saves Code Snippets, Ideas, To Do Lists...
- Store Account Info for Membership Sites & Web Mail
- Enhanced Web Favorites Organizer
- Export Your Personal Data to Text, XML & HTML
- Now Available for Classic Mac & Mac OS X

NEW! Save Calendars as HTML Pages

DOWNLOAD NOW
www.ebutterfly.com

electric butterfly

iScreensaver Designer

the cross-platform solution for Macintosh and Windows



Version 3.0 featuring OS X coming soon!

Mac designers: build professional screensavers without touching a Windows machine!

- build both Macintosh and Windows screensavers with a single click, no matter what system* you are using!
- use any QuickTime 6.0 movie format : Macromedia Flash 5.0, MPEG, Cinepak, MP3, Midi, AVI, DV Video... or, now with version 3.0, build your own basic Slide Shows!
- include a hidden movie that can be unlocked with a registration code
- customize and fully-brand both Installers and Screensaver control panels with pictures and text
- Screensavers install without DLLs, extensions, or restarts

simple WYSIWYG editor

supports interactive Flash and QuickTime

consistent cross-platform user interface

try before you buy fully functional online downloads



The iScreensaver Designer editing environment

creating screensavers for both Windows and Macintosh has never been this easy

<http://iScreensaver.net>
 email : info@iScreensaver.net

* supported systems, as of June 2002, include:
 Macintosh OS 8.6 to 9.2.2, Microsoft Windows 95/98/ME, NT4/NT2000/XP
 iScreensaver Designer version 3.0 will support up to Macintosh OS 10.2

©2000-2002 Xochi Media Inc. Made using REALbasic.

TitleTrack™ JUKEBOX

MAC MUSIC MANAGEMENT FOR SONY® 5 TO 400 DISC CD CHANGERS



Control up to 12 changers (4,800 music CDs)
 Control Any Brand Stereo Receiver
 Play MP3 and other Sound Files
 Plus much, much more!!!

www.titletrack.com info@titletrack.com


```

NSRect thisRect = NSIntersectionRect(
    [self rectOfColumn:col+i],
    [self rectOfRow:row]);
gridRect = NSUnionRect(gridRect,thisRect);
}
col += colspan;

// left edge. Only draw if this left edge isn't one we just drew.
leftEdge = (int) 0.5 + gridRect.origin.x;
if (leftEdge != oldLeftEdge)
{
    [NSBezierPath strokeLineFromPoint:
        NSMakePoint(
            -0.5+leftEdge, -0.5+gridRect.origin.y)
        toPoint:
            NSMakePoint(-0.5+leftEdge,
                -0.5+gridRect.origin.y
                + gridRect.size.height)];
}
// right edge
rightEdge = (int) 0.5 + gridRect.origin.x
    + gridRect.size.width;
[NSBezierPath strokeLineFromPoint:
    NSMakePoint(-0.5+rightEdge,
        -0.5+gridRect.origin.y)
    toPoint:
        NSMakePoint(-0.5+rightEdge,
            -0.5+gridRect.origin.y
            + gridRect.size.height)];
oldLeftEdge = rightEdge; // save edge for next pass through.
}
}
}
}

```

That's all there is to it. OK, maybe that one wasn't so easy. If you are going to be heavily subclassing NSTableView, realize that it is going to take a lot of trial and error, research, and help from other smart people to get it to work just right. Subclassing any object for which you don't have the source code is never easy, because you don't know all of the subtle interactions among the methods that you might be able to see if it was your code. Still, it is possible to modify NSTableView's default behavior by modifying just a few methods.

CUSTOM CELL CLASSES

That last one was a bit deep, so let's take a breather and try something a little easier. How do you make a "relevance" indicator, like you see when you search for items in the Finder? (See **Figure 5**.) As of Mac OS X 10.2, there is now a standard widget you need to use, but no such class has been provided in Cocoa for us to use. Since a Table displays cells, we need to make our own custom subclass of NSCell. We'll call the class "RankCell" because it's easier to spell than "RelevanceCell." (See **Listing 6**.)

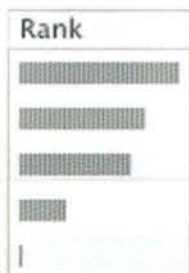


Figure 5: Relevance cells in a table

Listing 6: RankCell.m

```

// declare a static variable that will hold the pattern.
static NSImage *sRankPatternImage = nil;

```

initialize

When the class is initialized, load in the striped pattern from an image in the project.

```

+ (void) initialize
{
    sRankPatternImage
        = [[NSImage imageNamed:@"stripe"] retain];
}

```

floatValue

Return the value of the cell (a number from 0.0 to 1.0) as a floating point number. The method verifies that the cell's associated object value is indeed an object that can return a float. (Both NSNumber and NSString respond to floatValue)

```

- (float) floatValue
{
    float result = 0.0;
    id objectValue = [self objectValue];
    if ([objectValue respondsToSelector:
        @selector(floatValue)])
    {
        result = [(NSNumber *)objectValue floatValue];
    }
    return result;
}

```

setFloatValue:

Set the value of the cell to a number between 0.0 and 1.0.

```

- (void) setFloatValue:(float)inValue
{
    float value = inValue;
    if (value > 1.0) value = 1.0;
    if (value < 0.0) value = 0.0;
    [self setObjectValue:[NSNumber numberWithInt:float:value]];
}

```

drawInteriorWithFrame:inView:

Draw the cell. The cell will be smaller if its controlSize is NSSmallControlSize.

```

(void) drawInteriorWithFrame: (NSRect)inFrame
    inView: (NSView*)inView;
{
    float drawWidth;
    NSRect fillFrame, eraseFrame;

    // Constrain the frame's height
    float yInset
        = (NSSmallControlSize == [self controlSize])
            ? 4.0 : 3.0;
    NSRect newFrame = NSInsetRect(inFrame, 3.0, yInset);

    // Calculate width of filled part
    drawWidth
        = floor([self floatValue] * newFrame.size.width);
    if (drawWidth < 1)
    {
        drawWidth = 1; // at least 1 pixel wide, so we see something!
    }

    NSDivideRect(newFrame, &fillFrame, &eraseFrame,
        drawWidth, NSMinXEdge);

    [[NSColor colorWithPatternImage:sRankPatternImage] set];
    [NSBezierPath fillRect:fillFrame];
}

```

All we need to do is to set our table to use our RankCell class. As usual, a good place to do this is in an awakeFromNib method. (See **Listing 7**.) Then you have your data source's

tableView:objectValueForColumn:row: method return an NSNumber between 0.0 and 1.0. That's it!

Listing 7: CellDelegate.m

```
awakeFromNib
Create a RankCell and set it as a column's cell.

- (void)awakeFromNib
{
    RankCell *rankCell
    = [[RankCell alloc] initWithReuseIdentifier:@"rank"];

    NSTableColumn *rankColumn
    = [oTable tableColumnWithIdentifier:@"rank"];

    [rankCell setControlSize:NSSmallControlSize];
    [rankColumn setDataCell:rankCell];
}
```

ANIMATED SORTING

If you've seen iChat, you have probably noticed the slick animation when people on your Buddy List change status. The table elements animate as they move around. Very slick, but how did they do it? It probably involved subclassing NSTableView. Inspired by some Apple sample code called "AnimatedSlider" that animates the transition between values of an NSSlider (much like the preset equalizer settings in iTunes), I've determined out how to do this, and I present the technique here. (It's hard to show animations in print, but Figure 6 should give you an idea of what it looks like.)



Figure 6: Animated Sorting: Before, During, and After

The trick is to override the NSTableView method rectOfRow: and invoke it multiple times over a second or so, to move the rows from their old positions to their new ones. Unlike the "AnimatedSlider" example, which managed to put all the animation functionality into a category method on NSSlider, this is a little bit more complex, so we need the controller involved as well. So to keep the view separated as much as possible from the controller, the NSTableView subclass—AnimatedTableView—lets its delegate (or its data source, if you desire) take care of the logic associated with the animation. (See Listing 8.)

Listing 8: AnimatedTableView.m

```
rectOfRow:
Override NSTableView's method that calculates the rectangle of a given row. Let the
```

delegate determine what the rectangle for a given row is, if that delegate responds to the selector to do so. If animation is going on, it will probably return a value other than the default.

```
- (NSRect)rectOfRow:(int)rowIndex
{
    NSRect theDefaultRect = [super rectOfRow:rowIndex];
    if ([[self delegate] respondsToSelector:
        @selector(tableView:rectOfRow:defaultRect:)])
    {
        return [[self delegate]
            tableView:self rectOfRow:rowIndex
            defaultRect:theDefaultRect];
    }
    else
    {
        return theDefaultRect;
    }
}
```

unanimatedRectOfRow:
Return the default NSTableView's version of the rectangle for a given row; this method is provided so we can determine the rectangle as if animation weren't happening.

```
- (NSRect)unanimatedRectOfRow:(int)rowIndex
{
    return [super rectOfRow:rowIndex];
}
```

rowsInRect:
Override NSTableView's method that determines what rows are visible for a given rectangle. Let the delegate determine what the range of rows is for a given rectangle, if that delegate responds to the selector to do so. If animation is going on, it will probably return a value other than the default.

```
- (NSRange)rowsInRect:(NSRect)inRect
{
    NSRange theDefaultRange = [super rowsInRect:inRect];
    if ([[self delegate] respondsToSelector:
        @selector(tableView:rowsInRect:defaultRange:)])
    {
        return [[self delegate]
            tableView:self rowsInRect:inRect
            defaultRange:theDefaultRange];
    }
    else
    {
        return theDefaultRange;
    }
}
```

In order to animate the rows in a table, we need to keep track of both the original positions and the new positions of each row in the table! One way to do this is to build a new array of index values so we can lookup the original array position for any row in the sorted array. For example, if an array containing E, G, B, D, F is sorted into B, D, E, F, G, then we could build an array containing 2, 3, 0, 4, 1. Element 0 of the new array, B, used to be at position 2 in the old array; Element 1, D, used to be at position 3, and so forth. And what better way to implement this using a category method on NSArray! (See Listing 9.)

Listing 9: SortingDelegate.m

```
@interface NSArray (findPositions)
(NSArray *)
    findPositionsInUnsortedArray:(NSArray *)fromArray;
@end

@implementation NSArray (findPositions)

    findPositionsInUnsortedArray:
Build an array of NSNumbers, representing the position each item of an array used to be before it was sorted.
```



```

(NSArray *)
    findPositionsInUnsortedArray:(NSArray *)fromArray
{
    NSMutableArray *result
        = [NSMutableArray arrayWithCapacity:[self count]];
    NSEnumerator *theEnum = [self objectEnumerator];
    id object;

    while (nil != (object = [theEnum nextObject])) {
        int indexInOld = [fromArray indexOfObject:object];
        [result addObject:[NSNumber numberWithInt:indexInOld]];
    }
    return result;
}

@end

```

Now it's time for the actual animation, defined in `SortingDelegate.m`. (Note that this class contains three relevant instance variables: `mTimer`, a reference to the `NSTimer` used for sorting; `mOldPositionsArray`, the lookup array described above; and `mAnimationPosition`, a floating point value representing how far along in the animation we are.) Let's dive right into **Listing 10** for the methods we need.

Listing 10: `SortingDelegate.m`

```

//The "frame rate" for animating the table sort.
const float kFrameRate = 1.0/30;

// Duration of the animation, probably shouldn't be more than a second.
const float kAnimationTime = 0.75;

                                                                    setOldPositionsArray:
Set the array that holds the old positions of the items before they were sorted.

- (void) setOldPositionsArray:(NSArray *)inNewValue
{
    [inNewValue retain];
    [mOldPositionsArray release];
    mOldPositionsArray = inNewValue;
}

                                                                    stopAnimating
Remove any animating NSTimer. Also clears mAnimationPosition to indicate that we
are no longer animating.

- (void) stopAnimating
{
    [mTimer invalidate];
    mTimer = nil;
    mAnimationPosition = 0;
}

                                                                    setTimer:
Set the animation timer, replacing any existing one.

- (void) setTimer:(NSTimer *)inNewValue
{
    [inNewValue retain];
    [mTimer release];
    [self stopAnimating];
    mTimer = inNewValue;
}

                                                                    tableView:rowsInRect:defaultRange:
Invoked by AnimatedTableView. If we're currently animating, we just return the entire
range of all rows, so that all rows get drawn no matter where they are. (They will still
be clipped properly, but if we don't override this, then some rows may not be drawn.)
There are probably ways to make this more efficient, calculating which rows need to
be displayed, but as long as our table isn't too big, this should be fine.

- (NSRange)tableView:(AnimatedTableView *)inTableView
    rowsInRect:(NSRect)inRect
    defaultRange:(NSRange)inDefaultRange
{
    if (mAnimationPosition > 0)    // are we currently animating?

```

```

{
    return NSMakeRange(0, [oData count]); // just return all rows
}
else
{
    return inDefaultRange;
}
}

```

`tableView:rectOfRow:defaultRect:`
Invoked by `AnimatedTableView`. Returns a rectangle for the given row in the table. If we are animating, we calculate a rectangle to be some percentage of the way between the row's old rectangle and its new rectangle.

```

(NSRect)tableView:(AnimatedTableView *)inTableView
    rectOfRow:(int)inRowIndex
    defaultRect:(NSRect)inDefaultRect
{
    if (mAnimationPosition > 0) // are we currently animating?
    {
        // Get the rectangles of where the row originally was, and where it will end up.
        int oldPosition = [[mOldPositionsArray
            objectAtIndex:inRowIndex] intValue];
        NSRect oldR
            = [inTableView unanimatedRectOfRow:oldPosition];
        NSRect newR
            = [inTableView unanimatedRectOfRow:inRowIndex];

        // t will be our fraction between 0 and 1 of how far along the row should be.
        float t = mAnimationPosition;    // linear position based on time

        // Calculate a rectangle between the original and the final rectangles.
        NSRect newRect = NSMakeRect(
            NSMinX(oldR) + (t * (NSMinX(newR) - NSMinX(oldR))),
            NSMinY(oldR) + (t * (NSMinY(newR) - NSMinY(oldR))),
            NSWidth(newR), NSHeight(newR) );
        return newRect;
    }
    else
    {
        return inDefaultRect; // not animating, just return the standard value.
    }
}

```

`animateStep:`
Invoked by the `NSTimer` multiple times to handle the animation. Calculate the position we are based on the current time, and then either force a display of the table, or stop the animation.

```

- (void) animateStep:(NSTimer *)inTimer
{
    NSDate *start = [inTimer userInfo];
    NSTimeInterval elapsed
        = fabs([start timeIntervalSinceNow]);
    mAnimationPosition = MIN(1.0, elapsed / kAnimationTime);

    // Done yet? Allow a bit of "servo jitter" to allow for floating-point messiness
    if (fabs(mAnimationPosition - 1.0) <= 0.01)
    {
        [self stopAnimating];
        [oTable display];    // force a display of the table in its normal state
    }
    else
    {
        [oTable display];    // force a display of the table in its new state.
    }
}

```

`tableView:animateSortFromArray:toArray:`
Initiate an animated sort. We pass in the old array before the sort, and the new sorted array. It kicks off a timer to start the animation with the current date/time passed in as user info so it can keep track of how far along the animation it is.

```

(void)tableView:(NSTableView *)inTableView
    animateSortFromArray:(NSArray *)fromArray
    toArray:(NSArray *)toArray
{
    NSTimer *timer;
    NSRange visibleRows;

```



```

[self stopAnimating]; // Stop any existing sort animation
[self setOldPositionsArray:
    [toArray findPositionsInUnsortedArray:fromArray]];

timer =
    [NSTimer scheduledTimerWithTimeInterval:kFrameRate
     target:self
     selector:@selector(animateStep:)
     userInfo:[NSDate date] // store the starting date as user info
     repeats:YES];
[self setTimer:timer]; // store the timer so we can stop it when done.
}

```

Actually sort the data. This is an extension of the sortData method described in part 2 of this series. For this sample, if the application delegate method animateSorts returns YES, then kick off the sort. Otherwise, sort without animation. In either case, the table selection is preserved across the sort so that any rows selected before the sort will be properly selected in their new positions after the sort.

```

- (void) sortData
{
    SortContext ctxt=( mSortingKey, mSortDescending );
    NSSet *oldSelection
        = [self saveSelectionFromTable:oTable];

    if ([[[NSApp delegate] animateSorts]])
    {
        NSArray *originalOrderArray = [oData copy];
        [oData sortUsingFunction:ORDER_BY_CONTEXT
         context:&ctxt];
        [self restoreSelection:oldSelection toTable:oTable
         refresh:NO];

        [self tableView:oTable
         animateSortFromArray:originalOrderArray
         toArray:oData];
        // When done, the data should be displayed correctly.
    }
}

```

```

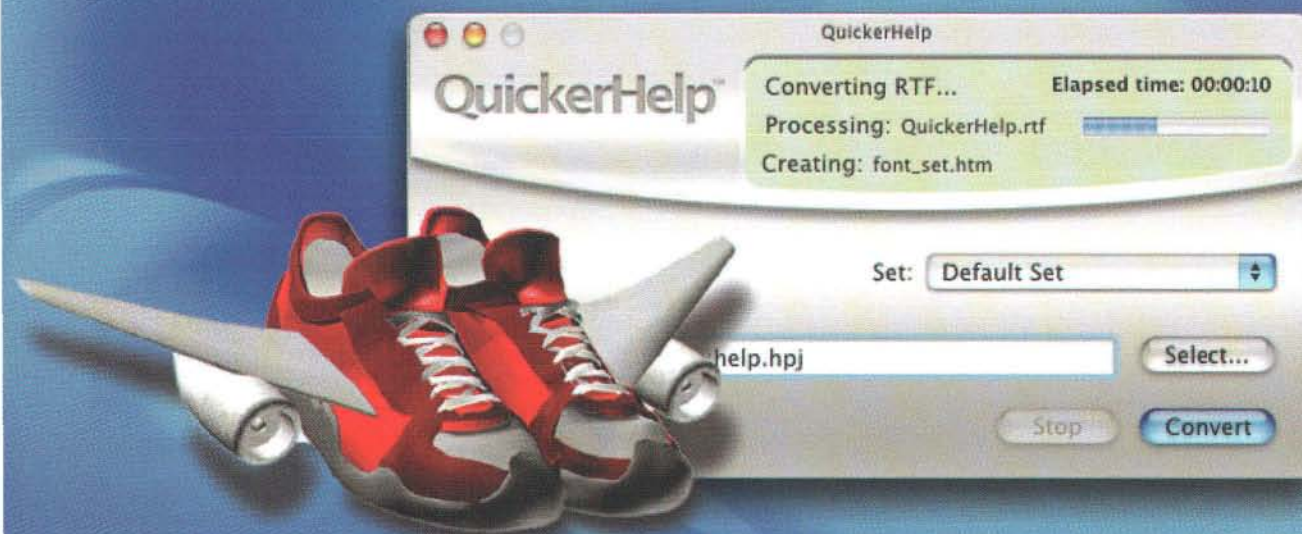
else
{
    [oData sortUsingFunction:ORDER_BY_CONTEXT
     context:&ctxt];
    [oTable reloadData];
    [self restoreSelection:oldSelection toTable:oTable
     refresh:YES];
}
}

```

That's all that's needed for a basic animated sort. You may find that you need to implement your own animation code differently to deal with your application's specific needs, but this should give you the basic idea. You are warned: This is not going to work very well on very large tables, since the position of every row in the table is being calculated repeatedly, regardless of whether it is visible or not. But for reasonable sizes, the animation can add a nice extra dimension to your interface.

There are a couple of visual improvements that can be made to the animation, however. The first improvement, demonstrated in the AnimatedSlider example, is making the movement seem more natural by letting the movement accelerate rather than abruptly changing from standing still to moving full-speed. Currently, the Y value (vertical position) changes linearly with respect to time, as depicted mathematically in **Figure 7**. It is possible to adjust the position by inserting a sine function into the calculations to ease the transition. **Listing 11** shows the function to convert a value between 0.0 and 1.0 into its eased

The fastest road to Apple Help starts here.



For moving your Help files to Apple Help, there's no faster, cleaner way than with QuickerHelp.
\$595 with discounts for ADC members. See www.mackiev.com/quickerhelp for details.



QuickerHelp™
The fastest road to Apple Help™



equivalent, depicted in **Figure 8**. All we need to do is insert `t = easeFunction(t);` into `tableView: rectOfRow: defaultRect:`.

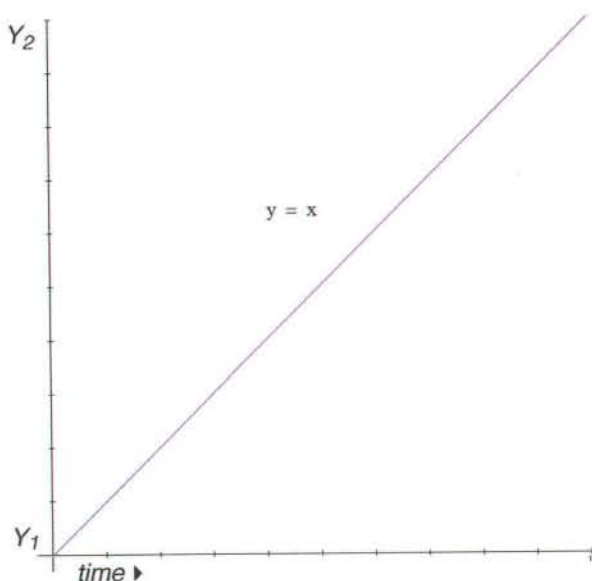


Figure 7: Linear Movement

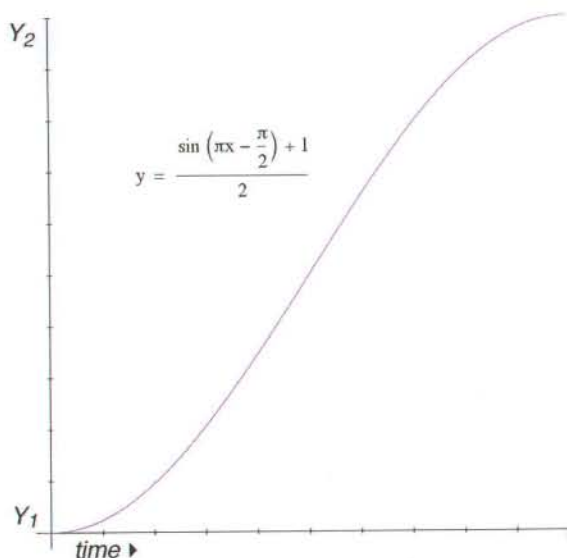


Figure 8: A Sine wave eases the transition.

Listing 11: SortingDelegate.m

easeFunction

This function implements a sinusoidal ease-in/ease-out for $t = 0$ to 1.0 . T is scaled to represent the interval of one full period of the sine function, and transposed to lie above the X axis.

```
float easeFunction(float t)
{
    return (sin((t * M_PI) - M_PI_2) + 1.0) / 2.0;
}
```

The other visual improvement that can be made is only apparent when you animate the reversal of a sort, changing from ascending to descending sort order, or vice-versa. Since the first items in the table exchange themselves symmetrically with the last items, they all end up bunching together in the center of the table halfway through the process. (See **Figure 9**.) It doesn't look as cool as it could. How can we make the movement a little bit less symmetrical so that everything doesn't crowd the middle of the table?

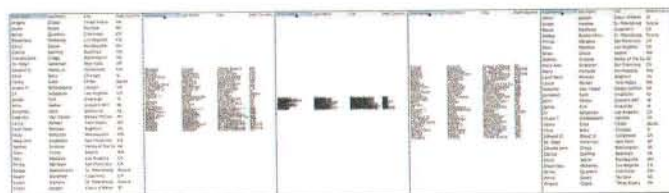


Figure 9: Reversing sort order bunches up in the middle.

If we could vary the "speed" of each row so that the rows at the top of the table move at a different rate than the rows at the bottom of the table, this would prevent the traffic jam in the center of the table. One way to do this is to use a "curve" function to move rows, with a different parameter used for each row. A reasonable function is shown in **Figure 9**. All we need to do is vary the power (n). Rows at one end of the table use a value slightly less than one; the middle row uses a power of one (a straight line); Rows at the other end of the table use a value slightly greater than one. We get a different curve for each row, and each row moves at a different rate.

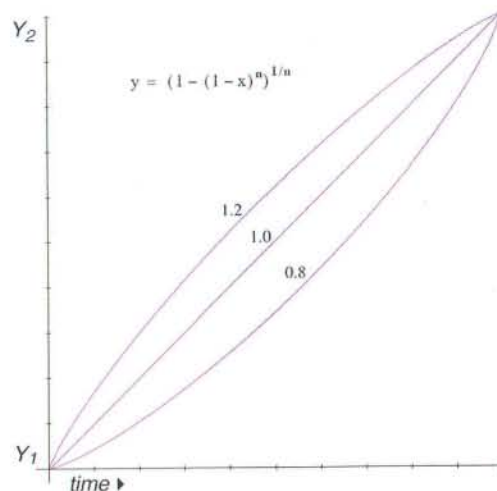


Figure 10: Power function with varying values of n

The effect of this is very nice; it looks as if the rows are doing a "back flip" as they swap their positions. **Figure 11** might give you a sense of how it looks, but it's more fun to see it in the program itself.

Watch and Record TV on your Mac!



Pause live television, skip commercials and record your favorite shows all on your Mac!

EL GATO
SOFTWARE

EYE TV
DIGITAL VIDEO RECORDER

Digital Media Remote



Works through USB to control multimedia applications on your Mac or PC computer.

FireWire/USB MicroGB

wiebeTECH



Holds up to 48GBs, is compatible with any laptop or desktop computer and fits in your pocket!



Dr. Bott

Attention Resellers: Dr. Bott has hundreds of useful, high-quality products for Macs!

877.611.2688 (toll free)

503.582.9944

www.drbott.com

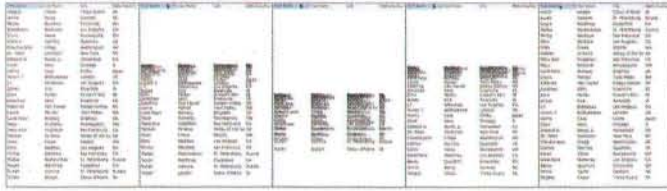


Figure 11: Adding a “curve” makes a reversal look nicer.

If you happen to be viewing only a piece of a table—say, the top half or the bottom half—and the rest is scrolled out of view, it can look pretty strange for the rows to move out of view and then move back into view a moment later. So we can perform an adjustment to the swapping behavior so that the curve is “centered” on the half of the table that you are looking at. So if you are viewing the top half of a table, the curve will be flipped compared to viewing the bottom half of a table.

Listing 12 shows the function you need to convert the “t” value into its curved counterpart.

Listing 12: SortingDelegate.m

// How “curvy” the movement should be. 0.2 or 0.3 is a nice value; 0.5 is funky!
const float kCurve = 0.3;

curveFunction

This function is used to de-center t (from 0 to 1.0) by a power p (a reasonable range is 0.8 to 1.2). It will make the animated reordering a little more interesting to watch.

```
float curveFunction (float t, float p)
{
    return pow( 1 - pow((1-t),p) , 1/p );
}
```

You will need to determine if the user is viewing only the top half of the table before you kick off the timer in tableView: animateSortFromArray: toArray:. This code just checks if the topmost visible row is less than 0.6 times the number of rows.

```
visibleRows = [inTableView rowsInRect:[inTableView
visibleRect]];
mViewingTop
    = NSMaxRange(visibleRows) < 0.6 * [fromArray count];
```

You need to insert the following lines into tableView: rectOfRow: defaultRect:

```
float rowPos = ((float) inRowIndex / [oData count]);
// fractional position of row in table
float rowPosAdjusted
    = mViewingTop ? (1.0 - rowPos) : rowPos;
float p = rowPosAdjusted * (kCurve*2.0) + 1.0 - kCurve;
// e.g. 0 > 0.8; n/2 -> 1.0; n -> 1.2
// (p actually could range from 1.0+kCurve to its reciprocal to be truly symmetrical)
t = curveFunction(t, p);
```

To bring this all together, we really want to have both the curve *and* the sinusoidal movement. So when we invoke curveFunction and then easeFunction together, we get movement as depicted in **Figure 12**. What more could you ask for?

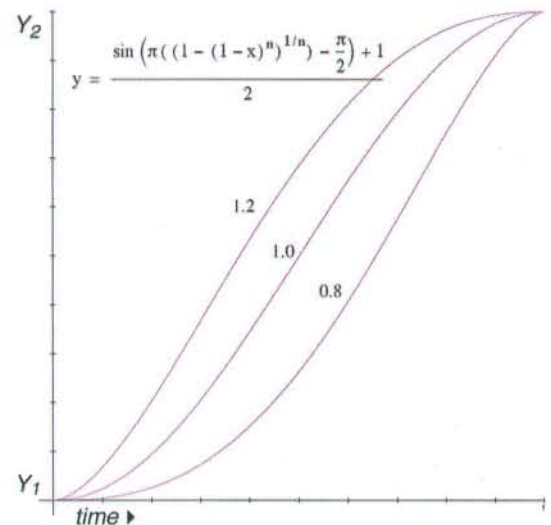


Figure 12: Combining the power and the sine functions

CONCLUSION

Whew! That has been a long trip through Table Land. Hopefully you feel armed with just about everything you need to make your application really shine. Quite a bit of code needs to be added to a simple NSTableView to get all the bells and whistles in place, but it's still possible to keep a clean separation between your views and your controllers. Accessing your data from the controllers is still quite simple thanks to Cocoa's Foundation Kit.

If you want to take advantage of many of these features, you will find yourself needing to implement several data source and delegate methods, and probably build a subclass for your table view that is an amalgamation of the subclasses presented here. An example subclass might be DeletableStripedTypeaheadAnimatedSortingTableView, but you are of course welcome to invent a shorter class name. (Never use a long word when you can use a diminutive one!)

FURTHER REFERENCE

Many of these techniques are discussed on the two dominant Cocoa discussion lists, at Apple (<http://lists.apple.com/mailman/listinfo/cocoa-dev>) and OmniGroup (<http://www.omnigroup.com/developer/maillinglists/macosx-dev/>). A great way to search the archives of these lists is to use <http://cocoa.mamasam.com>. Search the archives and post questions to these lists if you are attempting to do something with NSTableView that isn't discussed here.

The Web holds other resources related to NSTableView as well. NSTableView is discussed on cocoa-dev.com; Stone Design (<http://www.stone.com/dev/>) has a subclass available online; OmniGroup (<http://www.omnigroup.com/developer/sourcecode/>) has a number of extensions in their OmniAppKit code; Stéphane Sudre has some good NSTableView articles (in French) at <http://www.mosx.net/dev/>.

What's the point of driving a Jaguar if you can't get it out of first gear?

Mac® OS X v10.2, a.k.a. Jaguar, is truly a marvel of innovation and software engineering. The latest release of the Mac OS includes over 150 new features such as the Point-to-Point Tunneling Protocol (PPTP), Lightweight Directory Access Protocol 3.0, integration of FreeBSD 4.4 and GCC 3.1 into Darwin, and Zero Configuration Networking (Rendezvous®).

But having such a powerful machine on your desktop doesn't do much good if you don't know how to take full advantage of all that it has to offer.

Where the rubber meets the road.

In addition to covering the basics of Mac OS X, *Mac OS X: The Missing Manual*, 2nd Edition, has been fully updated to include all of Jaguar's new features. Serious power users and developers can get under the hood with chapters on connecting to Windows® networks, wireless networking, and an update of open source technologies.

If you're a Linux or Unix developer interested in the Mac OS, we've got everything you need to get into high gear. *Mac OS X for Unix Geeks* is your guide to figuring out the subtle differences between Mac OS X and other versions of Unix. And *Learning Cocoa with Objective-C* is for anyone developing applications for Mac OS X. It's the only book on the topic that's been reviewed and approved by Apple's own engineers.

Go ahead. Start your engines.
We've got you—and Mac OS X—covered.

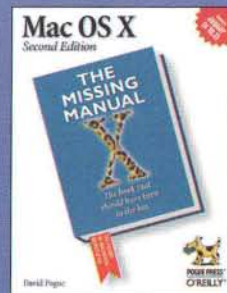


O'REILLY®

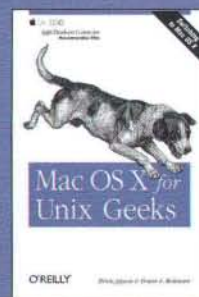
macdevcenter.com

800-998-9938

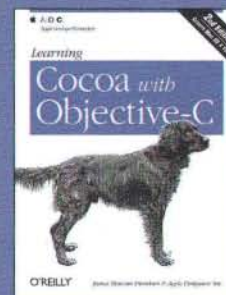
oreilly.com



**Mac OS X:
The Missing Manual, 2nd Edition**
David Pogue
ISBN 0-596-00450-8
\$29.95



Mac OS X for Unix Geeks
Brian Jepson & Ernest E. Rothman
ISBN 0-596-00356-0
\$24.95



**Learning Cocoa
with Objective-C, 2nd Edition**
James Duncan Davidson
& Apple Computer, Inc.
ISBN 0-596-00301-3
\$34.95

List of Advertisers

| | |
|--------------------------------------|-------|
| AEC Software | 33 |
| Aladdin Knowledge Systems, Inc. | 17 |
| Aladdin Systems, Inc. | 25 |
| Aladdin Systems, Inc. | 56 |
| Avesta Computer Services, Ltd. | 23 |
| Big Nerd Ranch, Inc. | 36 |
| Borland Software Corporation | 7 |
| DevDepot | 28-29 |
| Dr. Bott LLC | 77 |
| Einhugur Programming Resources | 70 |
| Electric Butterfly | 71 |
| Eudora Internet Mail Server | 56 |
| Exabyte | 21 |
| FairCom Corporation | 1 |
| Felt Tip Software | 23 |
| Fetch Softworks | 9 |
| Full Spectrum Software, Inc. | 19 |
| IDG World Expo Corporation | 50-51 |
| James Sentman Software | 70 |
| Karelia Software | 57 |
| Key3Media Group, Inc. / COMDEX | 62-63 |
| Lemke Software | 56 |
| Lingo Systems | 67 |
| MacDirectory | 37 |
| MacTech Magazine | 10 |
| Marx Software Security | 55 |
| Mathemaesthetics, Inc. | 35 |
| Microsoft | 53 |
| MYOB US, Inc. | 47 |
| /n software inc. | 39 |
| Netopia, Inc. | 41 |
| O'Reilly & Associates, Inc. | 79 |
| OpenBase International, Ltd. | 45 |
| Paradigma Software | 27 |
| Peachpit Press | 65 |
| Perforce Software, Inc. | 81 |
| piDog Software | 70 |
| PrimeBase (SNAP Innovation) | 59 |
| Prosoft Engineering, Inc. | 11 |
| REAL Software, Inc. | 69-71 |
| REAL Software, Inc. | 82 |
| RiverSong InterActive | 71 |
| Runtime Revolution Limited | ifc2 |
| Small Dog Electronics | 15 |
| Stone Design Corp | 42 |
| Sustainable Softworks | 58 |
| SyBase, Inc. | 2-3 |
| The Software MacKiev Company | 75 |
| Thursby Software Systems, Inc. | 13 |
| TLA Systems Ltd. | 57 |
| Trapcode Software | 57 |
| Trinfinity Software | 56 |
| USPS (US Post Office) | 31 |
| WIBU-SYSTEMS AG | 43 |
| Xochi Media Inc. | 71 |
| Xplain Corporation | 49 |
| Zero G Software | 61 |

List of Products

| | |
|--|-------|
| Accessories • DevDepot | 28-29 |
| Accessories • Dr. Bott LLC | 77 |
| Adobe Press • Peachpit Press | 65 |
| Big Nerd Ranch • Big Nerd Ranch, Inc. | 36 |
| Books • O'Reilly & Associates, Inc. | 79 |
| c-tree Plus • FairCom Corporation | 1 |
| COMDEX • Key3Media Group, Inc. / COMDEX | 62-63 |
| Consulting & Training Services • Avesta Computer Services, Ltd. | 23 |
| DAVE • Thursby Software Systems, Inc. | 13 |
| Development & Testing • Full Spectrum Software, Inc. | 19 |
| DragThing • TLA Systems Ltd. | 57 |
| EIMS • Eudora Internet Mail Server | 56 |
| FastTrack • AEC Software | 33 |
| Fetch • Fetch Softworks | 9 |
| GraphicConverter • Lemke Software | 56 |
| InstallAnywhere • Zero G Software | 61 |
| InstallerMaker, StuffIt • Aladdin Systems, Inc. | 25 |
| IP*Works! • /n software inc. | 39 |
| IPNetRouter & IPNetSentry • Sustainable Softworks | 58 |
| iScreensaver Designer • Xochi Media Inc. | 71 |
| JBuilder • Borland Software Corporation | 7 |
| MacDirectory • MacDirectory | 37 |
| MacTech Magazine Subscription • MacTech Magazine | 10 |
| Macworld Conference & Expo • IDG World Expo Corporation | 50-51 |
| Macworld Special Interest Pavilions • Xplain Corporation | 49 |
| MYOB • MYOB US, Inc. | 47 |
| Office for OS X • Microsoft | 53 |
| OpenBase • OpenBase International, Ltd. | 45 |
| piDog Utilities • piDog Software | 70 |
| Postal Permit Form • USPS (US Post Office) | 31 |
| PrimeBase • PrimeBase (SNAP Innovation) | 59 |
| Data Rescue • Prosoft Engineering, Inc. | 11 |
| QuickerHelp & Consulting • The Software MacKiev Company | 75 |
| REALbasic Plug-ins • Einhugur Programming Resources | 70 |
| REALbasic • REAL Software, Inc. | 82 |
| REALbasic Showcase • REAL Software, Inc. | 69-71 |
| Resorcerer • Mathemaesthetics, Inc. | 35 |
| Revolution • Runtime Revolution Limited | ifc2 |
| SCM Software • Perforce Software, Inc. | 81 |
| Security • Marx Software Security | 55 |
| SmallDog.com • Small Dog Electronics | 15 |
| Software and Hardware • Aladdin Knowledge Systems, Inc. | 17 |
| Software Protection • WIBU-SYSTEMS AG | 43 |
| Sound Studio • Felt Tip Software | 23 |
| Stone Studio • Stone Design Corp | 42 |
| SyBase • SyBase, Inc. | 2-3 |
| Ten for X • Aladdin Systems, Inc. | 56 |
| Timbuktu Pro & netOctopus • Netopia, Inc. | 41 |
| Time Track • Trinfinity Software | 56 |
| TitleTrack Jukebox • RiverSong InterActive | 71 |
| Translation & Localization • Lingo Systems | 67 |
| Trapcode • Trapcode Software | 57 |
| UniHelp Module • Electric Butterfly | 71 |
| Valentina • Paradigma Software | 27 |
| VXA • Exabyte | 21 |
| Watson • Karelia Software | 57 |
| Whistle Blower • James Sentman Software | 70 |

The index on this page is provided as a service to our readers. The publisher does not assume any liability for errors or omissions.

To meet deadlines, developers have two choices:

1. Use Perforce
2. Cut Corners



For developers under pressure to manage source code and do more in less time, Perforce's Fast Software Configuration Management System is the must-have tool.

With rival SCM systems, the only way to quicken the pace is to cut corners - but in the long run you pay the price with missed deadlines, uncertain contents, buggy releases and no way back to previous builds.

With Perforce, the fast way is always the right way. Install it fast, learn it fast, execute operations fast. With other SCM systems, developers face an unpleasant choice: do it the right way or do it the fast way. Perforce's speed and reliability mean fast is right. See how Perforce compares with other leading SCM systems at <http://www.perforce.com/perforce/reviews.html>

Run at full speed even with hundreds of users and millions of files. At the core of Perforce lies a relational database with well-keyed tables, so simple operations can be accomplished in near-zero time. Larger operations (like labeling a release and branching) are translated into keyed data access, giving Perforce the scalability that big projects require.

Work anywhere. Perforce is efficient over high-latency networks such as WANs, the Internet and even low-speed dial-up connections. Requiring only TCP/IP, Perforce makes use of a well-tuned streaming message protocol for synchronizing client workspace with server repository contents.

Develop and maintain multiple codelines. Perforce Inter-File Branching™ lets you merge new features and apply fixes between codelines. Smart metadata keeps track of your evolving projects even while they develop in parallel.

Truly cross platform. Perforce runs on more than 50 operating systems, including Windows and nearly every UNIX® variation, from Linux® and Mac OS® X to AS/400 and more.

Integrate with leading IDEs and defect trackers: Visual Studio.NET®, Visual C++®, Visual Basic®, JBuilder®, CodeWarrior®, TeamTrack®, Bugzilla™, ControlCenter®, DevTrack® packages, and more.

PERFORCE
SOFTWARE

Fast Software Configuration Management www.perforce.com

Download your free 2-user, non-expiring, full-featured copy now from www.perforce.com
Free (and friendly) technical support is on hand to answer any and all evaluation questions.

All trademarks used herein are either the trademarks or registered trademarks of their respective owners.

like ships
passing in
the night

ARE YOU HONEST, handsome, successful, financially secure, intelligent, world-traveled, cultured, creative, fun, playful, adventurous, passionate, humorous, caring, loving, and between 46 and 58? Respond to European blonde female correspondent. #C882

RECENTLY PAROLED, looking for a lady who will keep me on the straight and narrow. Must be into drugs and shoplifting. #6357

Come see us at Macworld in MacTech Central! Download a free demo. www.realbasic.com